COVID-19 New Confirmed Case Analysis and Forecast

Kelly Wang

6/5/2020

Abstract

At the beginning of year 2020, a pandemic disease named COVID-19 started to spread all over the world. Around 6.42 million people confirmed with the disease and 383 thousand people died until today. Unfortunately, the numbers are still increasing. Under the severe situation, this project intends to forecast the new confirmed cases of COVID-19 in the world in order to provide some useful information related to COVID-19 for the unknown future.

The project is based on time series analysis techniques including stationarity transformation, model selection, diagnostic checking and forecast. The final model chosen to do forecast is an ARIMA(1, 1, 6) model. The model past most of the tests and provided an effective forecast to the future situation of COVID-19.

Introduction

The dataset came from public resources of European Centre for Disease Prevention and Control (ECDC). It recorded the daily new confirmed COVID-19 cases of the world and every country separately. This project based on ECDC data is designed to forecast the future new confirmed cases of COVID-19 in the world. The number of COVID-19 cases is still increasing. If we know better about the upcoming sitution, we can take actions ahead of time to decrease the number of people infected or prepare more completely to face the new confirmed cases.

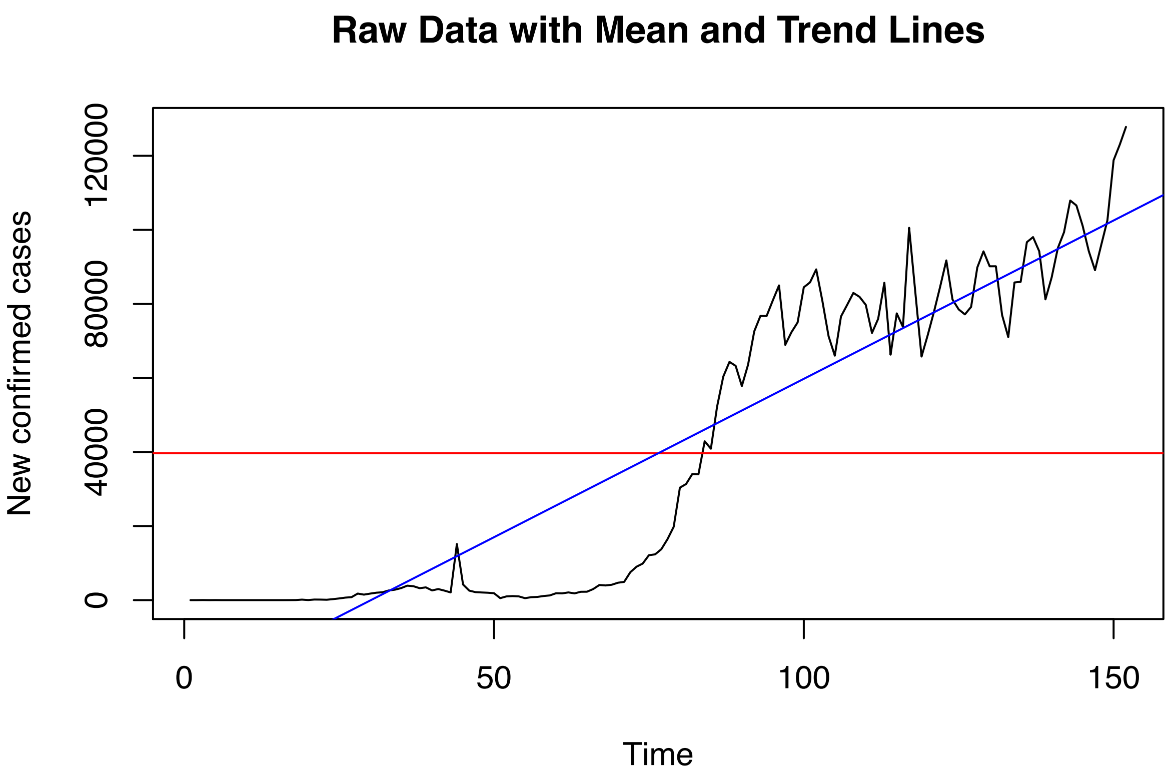
Instead of using the whole dataset, I extracted only the numbers of new confirmed cases of the world and period from 2020-01-01 to 2020-05-31 which contains 153 observations. The dataset was split into training data and test data by ratio 140:13. Since the dataset has non-constant variance and trend, I used Box-Cox transformation and differencing at lags to stablize the dataset. I selected several models based on ACF, PACF visualization and lowest order of AIC, and ended up with final two models after checking the stationarity and invertibility, which were an ARIMA(1, 1, 6) model and an ARIMA(0, 1, 8) model. The diagnostic checking showed that arima(1, 1, 6) model was better than arima(0, 1, 8) model. Therefore, ARIMA(1, 1, 6) $(1+0.9575B_{(0.0411)})(1-B)X_t = (1+0.5265_{(0.5265)}B-0.3411_{(0.0692)}B^2+0.4171_{(0.1059)}B^5+0.4931_{(0.1043)}B^6)Z_t$ became the final model that was used to do forecast.

The data had relative constant variance and was de-trended after transformation and differencing. Besides, the models I selected had low AIC and satisfied stationarity and invertibility. The residuals of models past all tests except normality test since the original data is very non-normal. The model forecasted a similar trend with real values though the forecast points were not exactly the same with true points. Therefore, the final model is $(1+0.9575B_{(0.0411)})(1-B)X_t = (1+0.5265_{(0.5265)}B-0.3411_{(0.0692)}B^2+0.4171_{(0.1059)}B^5+0.4931_{(0.1043)}B^6)Z_t$.

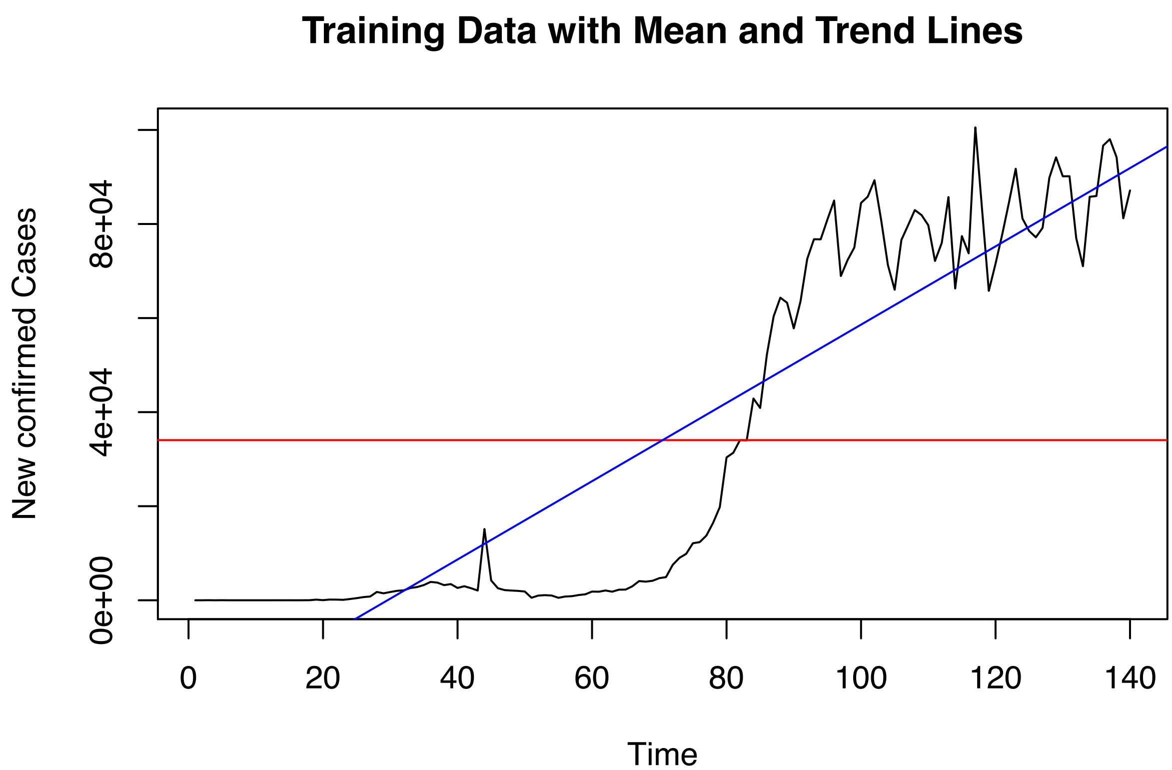
The dataset is from ECDC orginally and I extracted it from https://github.com/owid/covid-19-data/blob/master/public/data/ecdc/new_cases.csv. The software used for this project is RStudio.

Sections

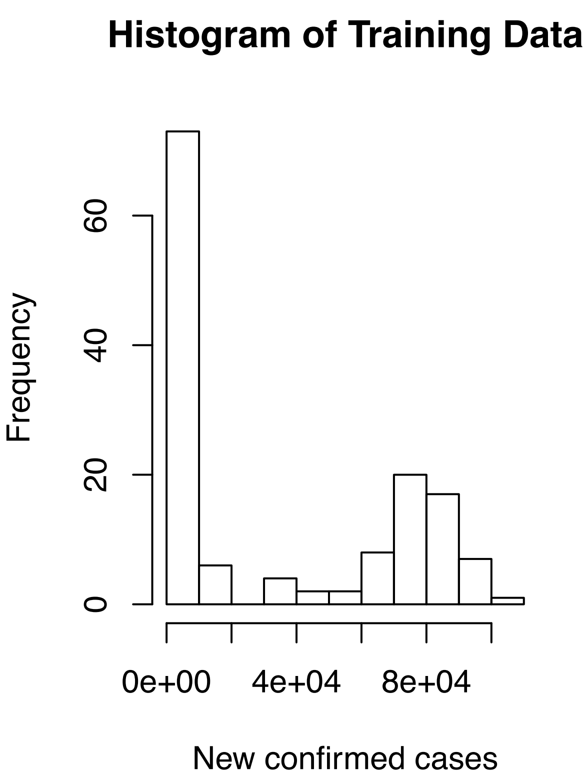
Preprocessing and Plot

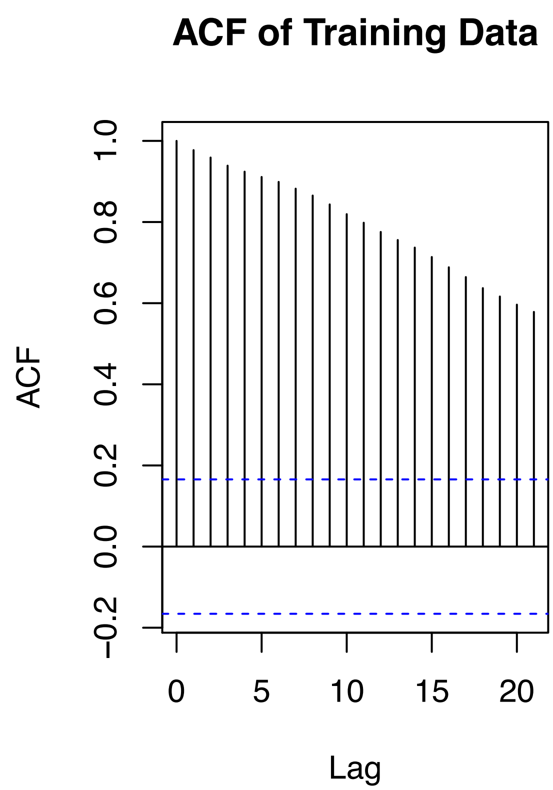


The time series is obviously non-stationary and mean is 40,000. It has a clear increasing trend and the variance is not constant over time. There exists a sharp increase starting around from 50 to 100, which is the starting from the end of February to the beginning of April. The sharp increase represented the breakout of COVID-19. After that, the increasing rate slightly decreased and the number of new confirmed cases started to bounce up and down frequently.



The training data has the same characteristics with original time series after splitting the orginal time series into training data and test data.

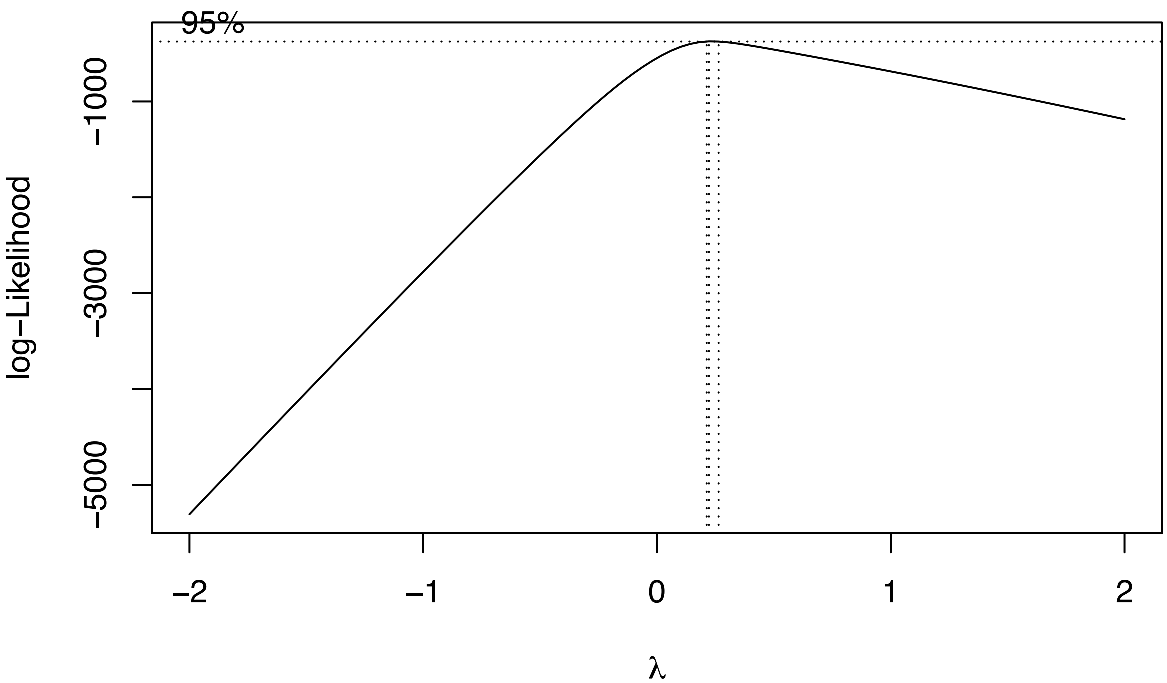




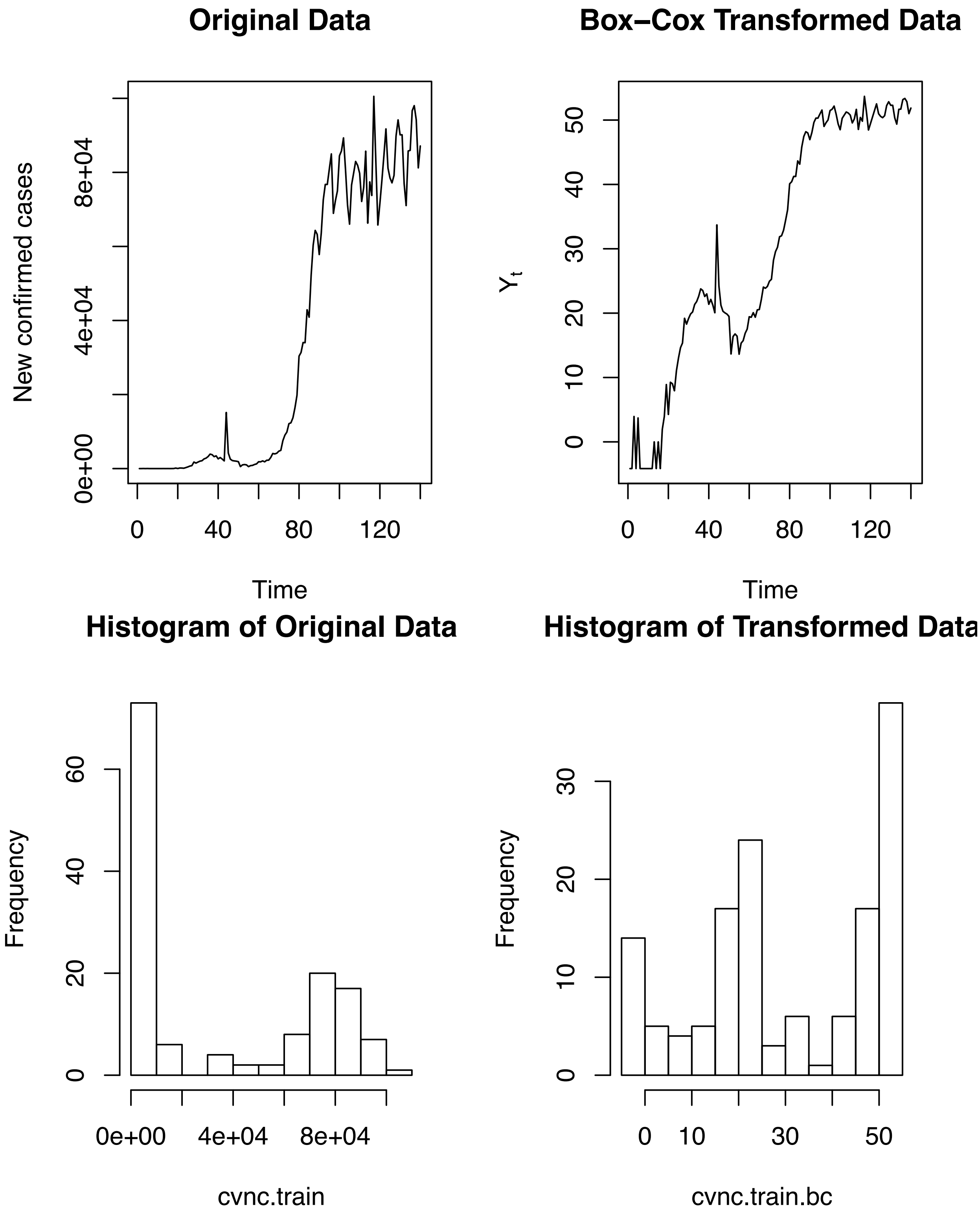
The histogram is badly skewed and acf remains large for training data. These two plots help confirm the non-staionarity.

Transformation

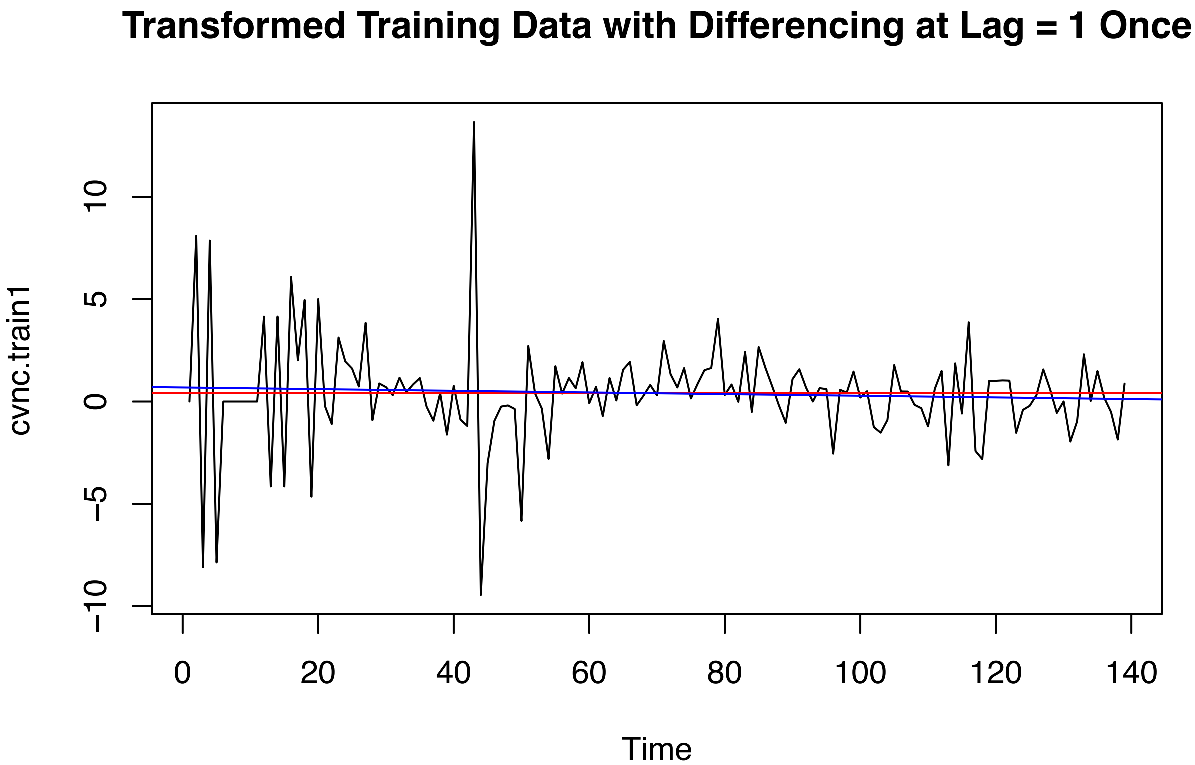
Since the time series has non-constant variance and trend, we need to apply some transformations for the time series.
To stablize the variance and skewed data, we will aplly the Box-Cox transformation.



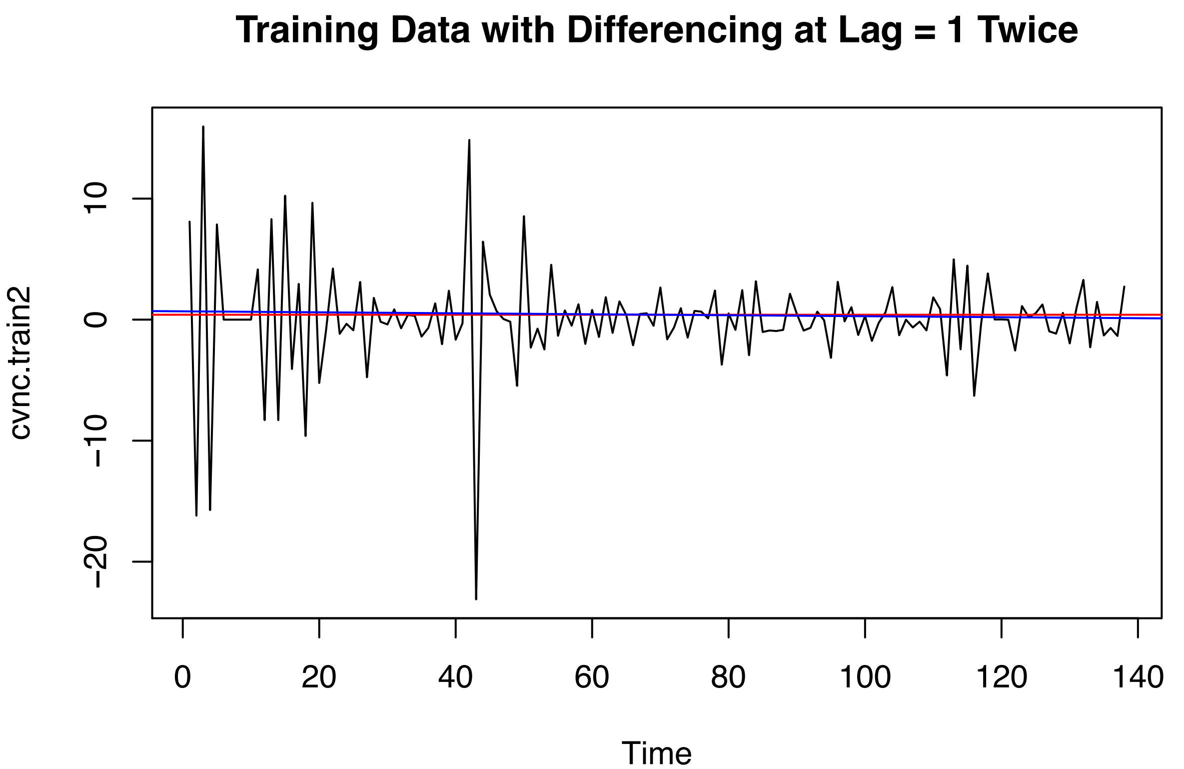
The most proper lambda is 0.2222222 and the 95% confidence interval does not contain 0. Thus, we will use the exact lambda that Box-Cox transformation function gives us to transform training data.



After applying Box-Cox transformation, the variance is more stable and the histogram is more symmetric. However, the trend sill exsits. In order to remove the trend, we will apply differencing at lag = 1 to the transformed data.



After differencing at lag = 1 once, the trend line is almost horizontal.



After differencing at lag = 1 twice, the trend lie is more horizontal.

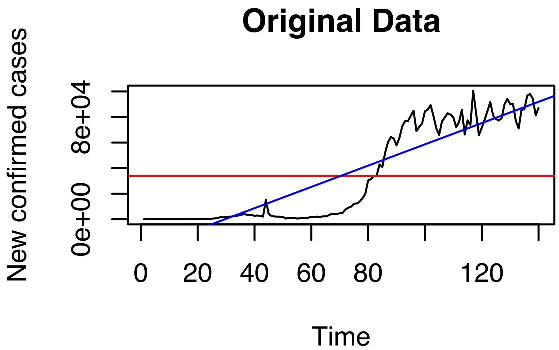
```
## [1] "Variance after differencing once:"
```

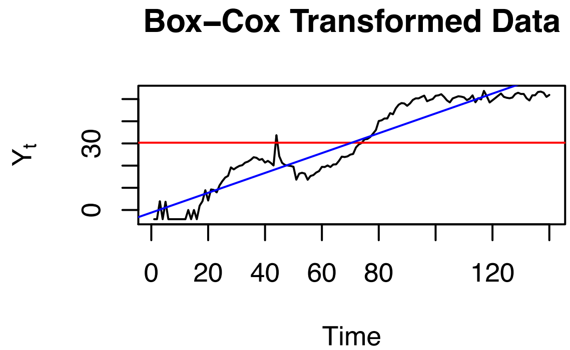
```
## [1] 6.974256
```

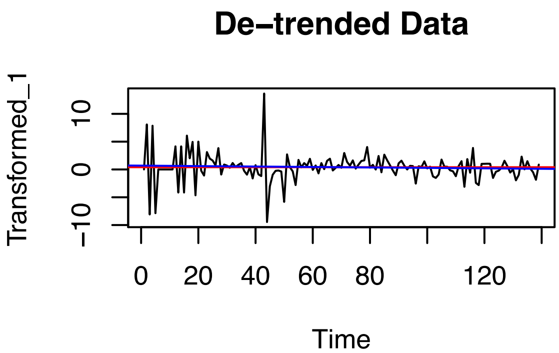
```
## [1] "Variance after differencing twice:"
```

```
## [1] 20.26478
```

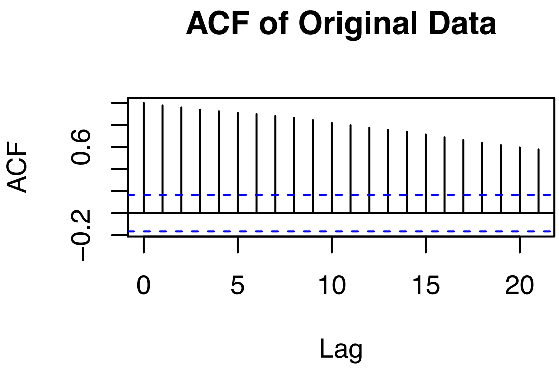
Since the variance with differencing at lag = 1 twice is higher than differencing once, we end up with applying differencing at lag = 1 once to the transformed data.

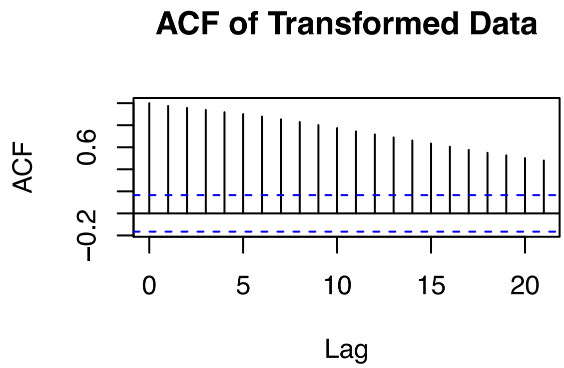


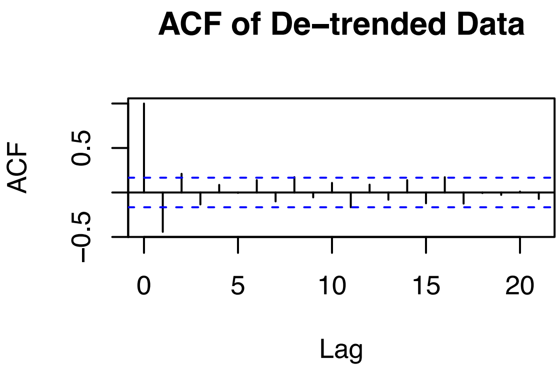




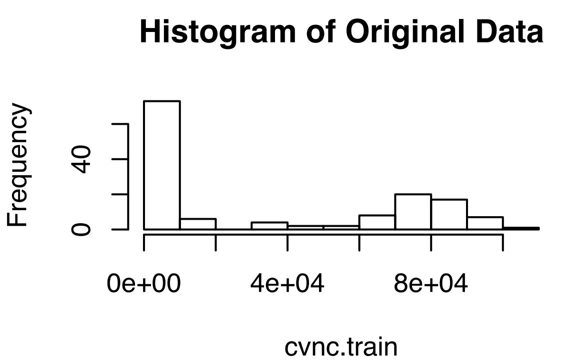
After applying Box-Cox transformation and differencing at lag = 1 once, variance is more stable and the trend is removed. The plot of de-trended transformed data looks stationary now.

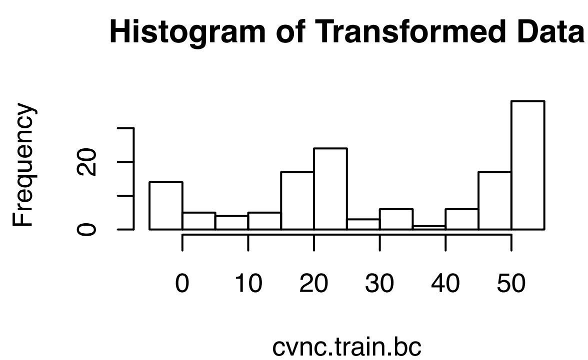


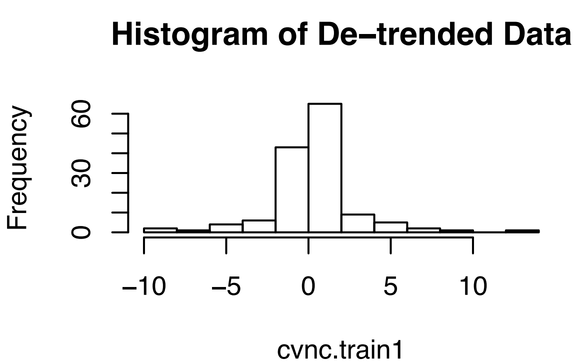




ACF remains large for the original data and transformed data. After applying the differencing, ACF decay corresponds to a stationary process.

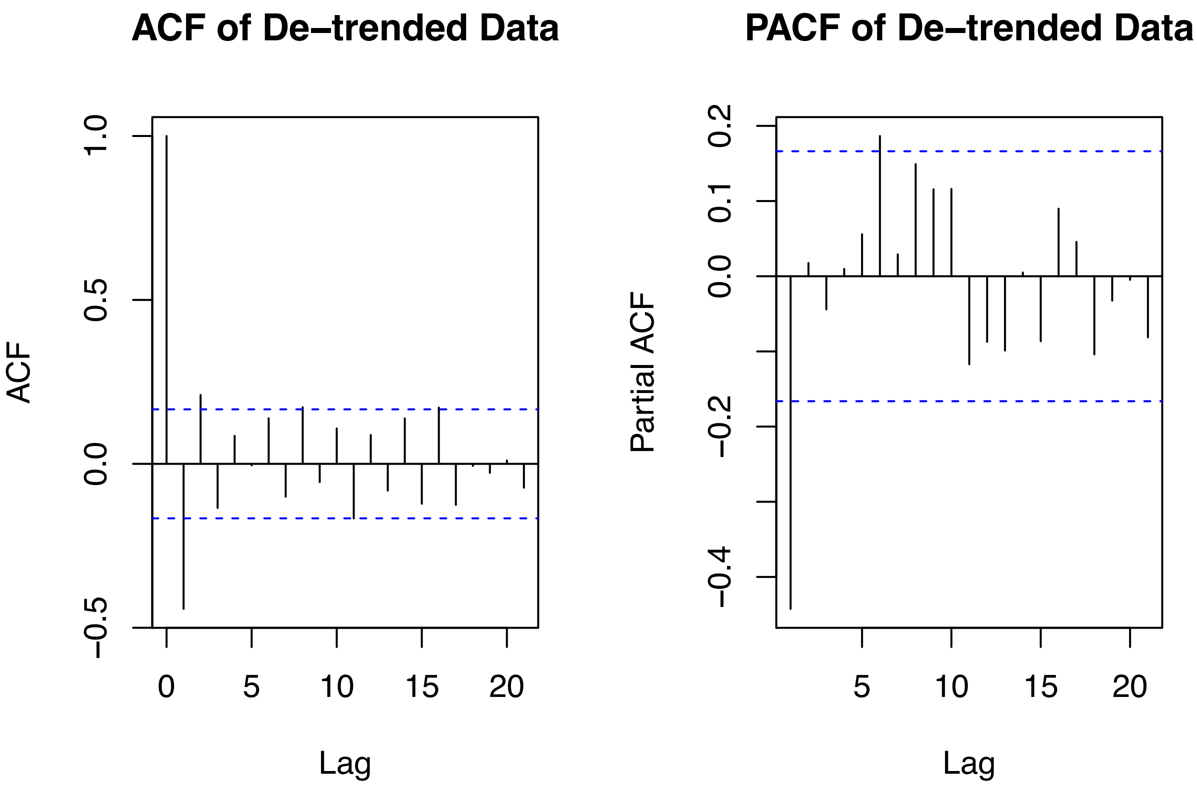






The histogram is getting more and more symmetric during the transformation.

Model Identification



ACF outside confidence intervals: lags 1, 2, 8, 11 and 16
PACF outside confidence intervals: lags 1 and 6
List of ARIMA models to try: d = 1; q = 2, 8, 11 or 16; p = 1 or 6
Order of moving average part q is determined based on ACF of transformed data and order of autoregressive part p is determined based on the PACF of transformed data. ACF at lags = 1 and 2 are out of confidence interval and ACF at lags = 8, 11 and 16 are on the confidence interval. PACF at lags = 1 and 6 are out of confidence interval. Therefore, 1, 2, 8, 11 and 16 are potential options for q and 1 and 6 are potential options for p.

Model Fitting

After trying all potential models, two models were selected based on 1) lowest AIC, 2) principle of parsimony and 3) stationarity and invertibility.

```
# Model 1
arima(cvnc.train.bc, order=c(1,1,8), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 8), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2     ma3     ma4     ma5     ma6     ma7
##       -0.9571  0.5208  -0.3097  0.1222  0.0890  0.3881  0.4522  0.0865
## s.e.   0.0408  0.1008   0.1002  0.1030  0.1122  0.1110  0.1179  0.1137
##          ma8
##       0.1069
## s.e.  0.1070
```

```
##
## sigma^2 estimated as 4.909:  log likelihood = -309.28,  aic = 638.56
```

The coefficients for ma3, ma4, ma7 and ma8 are not significant since they conclude 0 in confidence intervals. Therefore, we will fix them one by one and check the change of AIC after each fixing. If the AIC is higher after one fixing, we will keep the coefficient even if it is not significant.

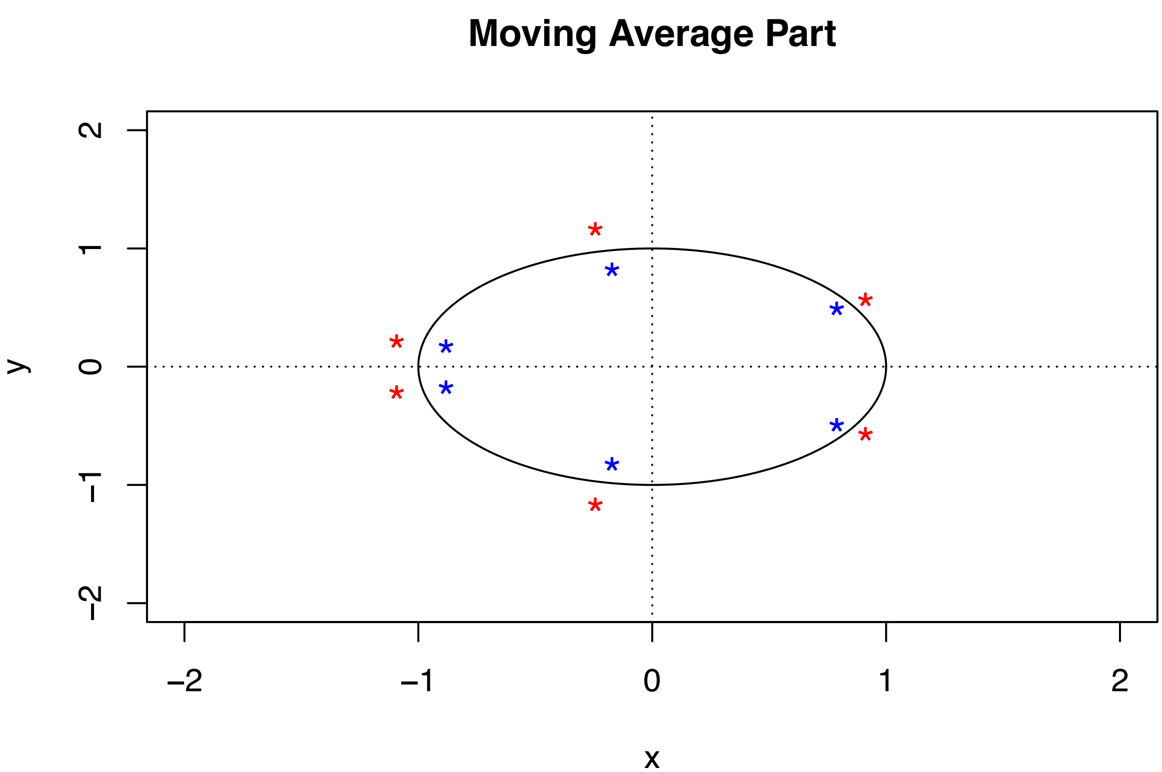
```r
# Fix coefficients
arima(cvnc.train.bc, order=c(1,1,6), method="ML", fixed = c(NA, NA, NA, 0, 0, NA, NA))
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 6), fixed = c(NA, NA, NA, 0, 0, NA,
##     NA), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2  ma3  ma4     ma5     ma6
##       -0.9575  0.5265  -0.3411    0    0  0.4171  0.4931
## s.e.   0.0411  0.0842   0.0692    0    0  0.1059  0.1043
##
## sigma^2 estimated as 4.981:  log likelihood = -310.15,  aic = 632.29
```

All insignificant coefficients are fixed to 0 and original ARIMA(1, 1, 8) model becomes ARIMA(1, 1, 6) model. AIC becomes 632.29 which is lower than orginal AIC 638.56. Therefore, the specific model is: $(1+0.9575B_{(0.0411)})(1-B)X_t = (1+0.5265_{(0.0842)}B-0.3411_{(0.0692)}B^2+0.4171_{(0.1059)}B^5+0.4931_{(0.1043)}B^6)Z_t$.

Since $|\phi_1| < 1$, the model is stationary already. We will only check invertibility for the model.

```r
# Check invertibility
source("plot.roots.R")
plot.roots(NULL, polyroot(c(1, 0.5265, -0.3411, 0, 0, 0.4171, 0.4931)), main="Moving Average Part")
```



The model is stationary becasue all roots are outside the unit circle.

```r
# Model 2
arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5     ma6      ma7     ma8
##       -0.4727  0.1770  -0.0805  0.1679  0.2477  0.1896  -0.0711  0.0846
## s.e.   0.0884  0.1053   0.1027  0.1043  0.1180  0.0973   0.1118  0.1142
##
## sigma^2 estimated as 5.135:  log likelihood = -311.87,  aic = 641.74
```

The coefficients for ma2, ma3, ma4, ma6, ma7 and ma8 are not significant since they conclude 0 in confidence intervals. We will try to fix them using the same procedure of fixing coefficients with model 1.

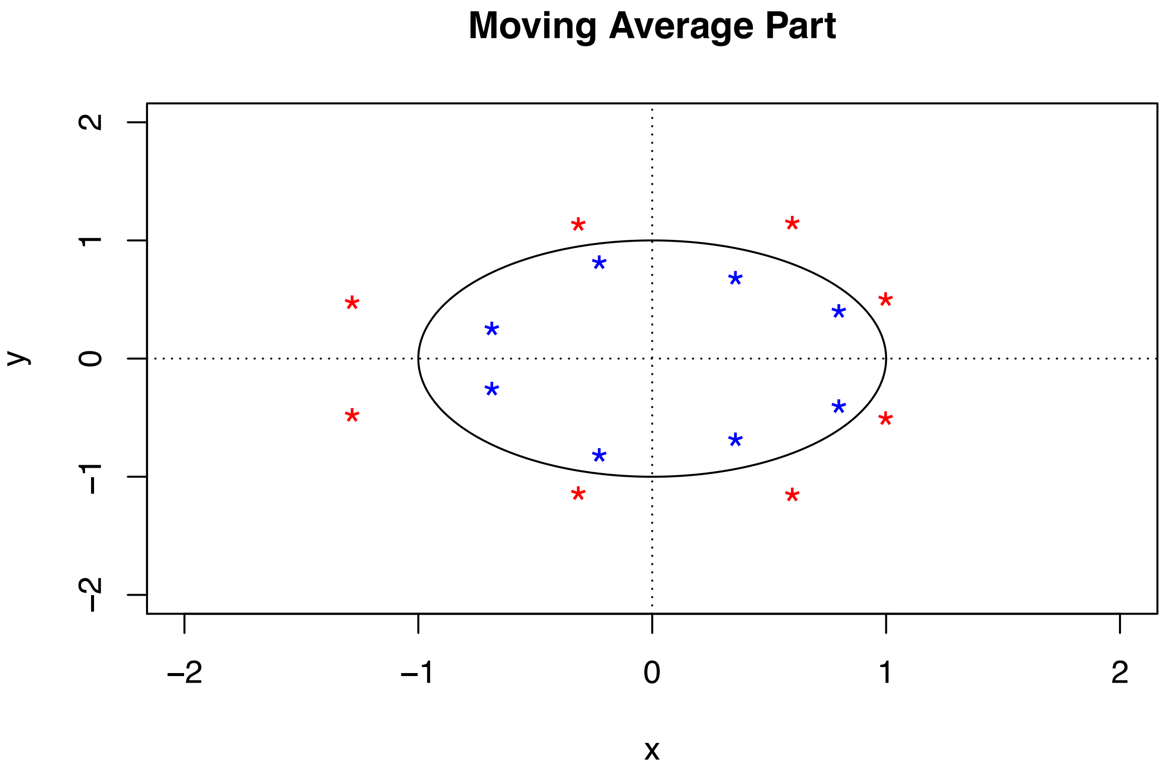
```r
# Fix coefficients
arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML", fixed = c(NA, NA, 0, 0, NA, 0, 0, NA))
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 8), fixed = c(NA, NA, 0, 0, NA, 0,
##     0, NA), method = "ML")
##
## Coefficients:
##           ma1     ma2  ma3  ma4     ma5  ma6  ma7     ma8
##       -0.4873  0.1936    0    0  0.2386    0    0  0.1819
## s.e.   0.0768  0.0903    0    0  0.1245    0    0  0.0850
##
## sigma^2 estimated as 5.312:  log likelihood = -313.78,  aic = 637.57
```

Different with model 1, only coefficients for ma3, ma4, ma6 and ma7 are fixed to 0 since setting coefficients for ma2 and ma8 will increase the AIC. Therefore, we keep the coefficients of ma2 and ma8 without fixing. The specific model is:
$(1-B)X_t = (1 - 0.4873_{(0.0768)}B + 0.1936_{(0.0903)}B^2 + 0.2386_{(0.1245)}B^5 + 0.1819_{(0.0850)}B^8)Z_t.$

Since the model is pure moving average model, we will only check invertibility for the model.

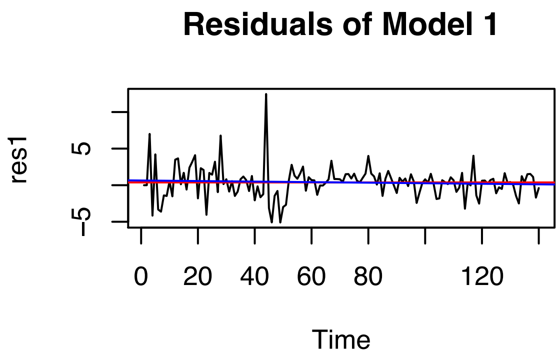
```r
# Check invertibility
plot.roots(NULL, polyroot(c(1, -0.4873, 0.1936 , 0, 0, 0.2386, 0, 0, 0.1819)), main="Moving Average Par
```

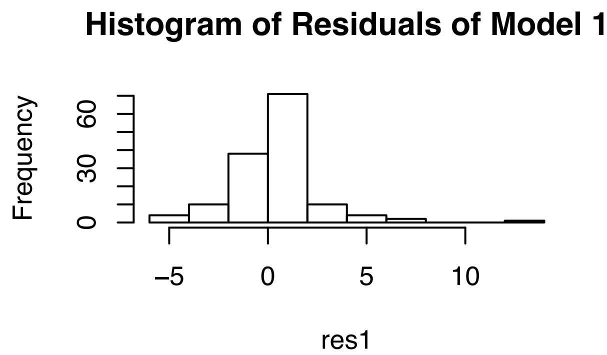


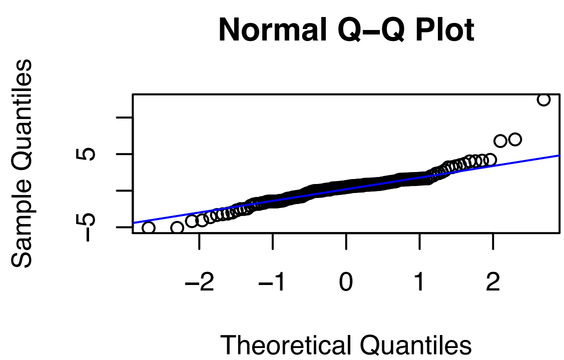
The model is stationary becasue all roots are outside the unit circle.

Therefore, the models we select are:
Model 1:
$(1+0.9575B_{(0.0411)})(1-B)X_t = (1+0.5265_{(0.0842)}B-0.3411_{(0.0692)}B^2+0.4171_{(0.1059)}B^5+0.4931_{(0.1043)}B^6)Z_t$
$\sigma^2 = 4.981$
Model 2:
$(1-B)X_t = (1-0.4873_{(0.0768)}B+0.1936_{(0.0903)}B^2+0.2386_{(0.1245)}B^5+0.1819_{(0.0850)}B^8)Z_t$
$\sigma^2 = 5.312$

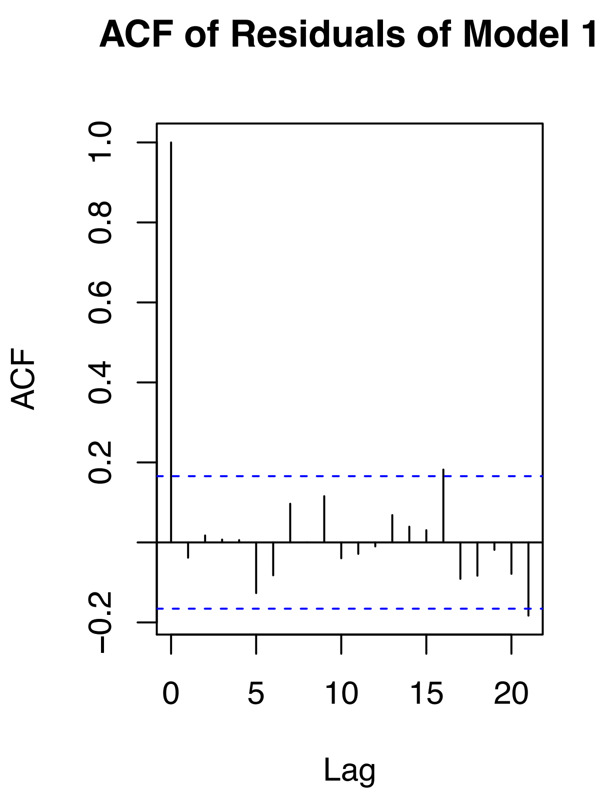
Diagnostic Checking

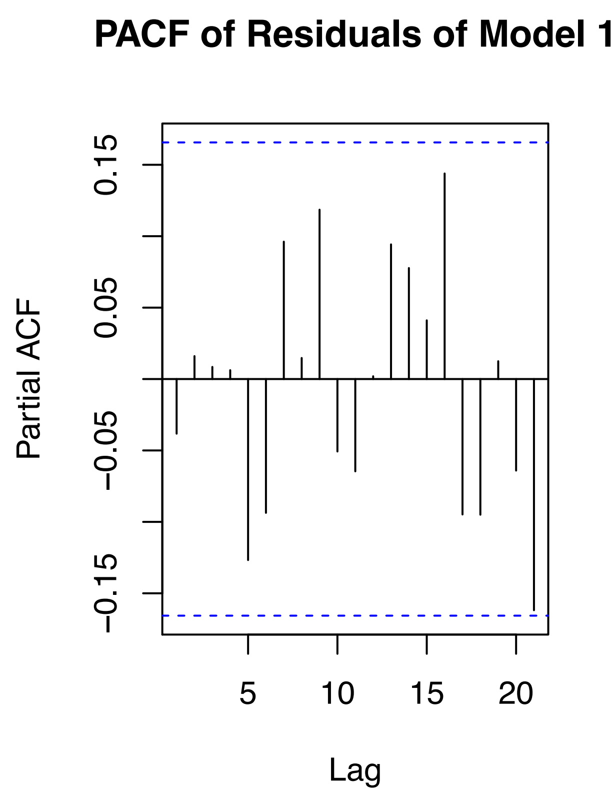






Sample mean is 0.3789728. From the plots for residuals, we can see that there is no trend or seasonality and the variance is stable. The histogram is symmetric. The Q-Q plot does not look very good.





The ACF of lag = 16 is slightly outside the confidence interval but we can disregard it by Bartlett's formula. Therefore, all the ACF and PACF are within the confidence interval and can be counted as 0.

```
# Residual tests for residuals of model 1
shapiro.test(res1)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res1
## W = 0.90403, p-value = 5.272e-08
```

```
Box.test(res1, lag = 12, type = c("Box-Pierce"), fitdf = 5)
```

```
##
##  Box-Pierce test
##
## data:  res1
## X-squared = 6.9895, df = 7, p-value = 0.43
```

```
Box.test(res1, lag = 12, type = c("Ljung-Box"), fitdf = 5)
```

```
##
##  Box-Ljung test
##
## data:  res1
## X-squared = 7.4482, df = 7, p-value = 0.3837
```

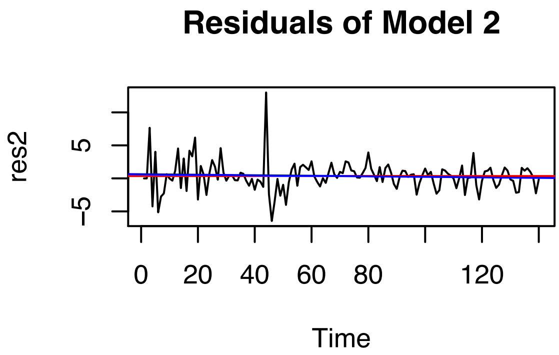
```
Box.test(res1^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
```

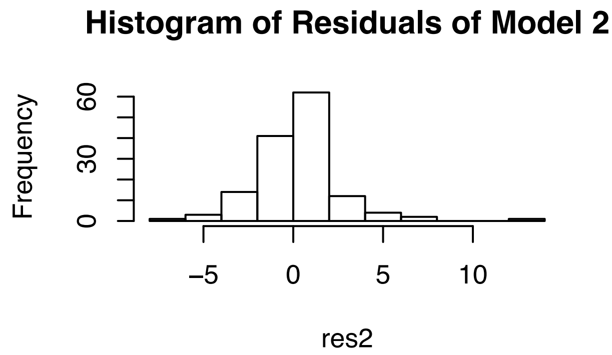
```
##
##  Box-Ljung test
##
## data:  res1^2
## X-squared = 5.7793, df = 12, p-value = 0.9268
```

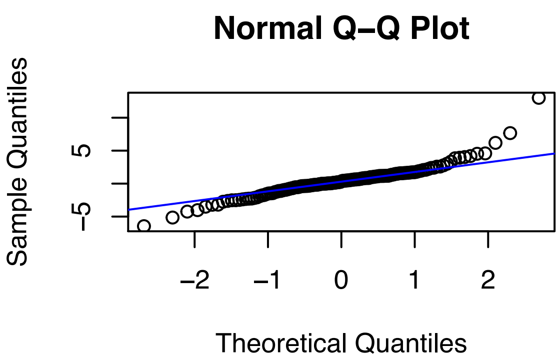
```
ar(res1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  4.837
```

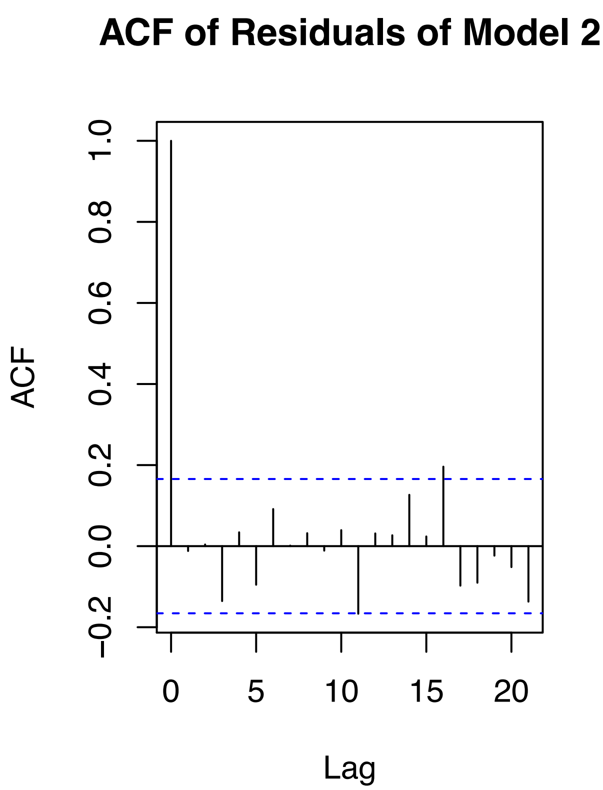
The residuals pass all the tests except Shapiro-Wilk normality test. The failure of passing Shapiro-Wilk normality test and Q-Q plot can be explained by the serious non-normality of the original time series. We have already applied transformation to original time series and it is hard for us to make further modification. Therefore, we will keep the model as $(1+0.9575B_{(0.0411)})(1-B)X_t = (1+0.5265_{(0.0842)}B - 0.3411_{(0.0692)}B^2 + 0.4171_{(0.1059)}B^5 + 0.4931_{(0.1043)}B^6)Z_t$ and conclude that the model is satisfactory.

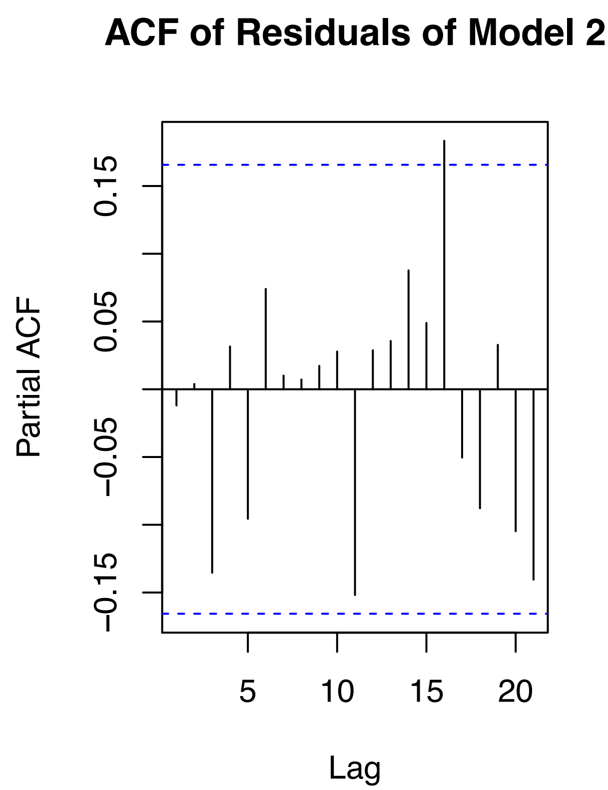






The sample mean is 0.3530809. From the plots we can see that there is no trend or seasonality and the variance is stable. The histogram is symmetric. However, the Q-Q plot does not look very good.





ACF at lag = 11 is around confidence interval and ACF at lag = 16 is outside confidence interval. PACF at lag = 16 is outside confidence interval.

```r
# Residual tests for residuals of model 2
shapiro.test(res2)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res2
## W = 0.90628, p-value = 7.064e-08
```

```r
Box.test(res2, lag = 12, type = c("Box-Pierce"), fitdf = 4)
```

```
##
##  Box-Pierce test
##
## data:  res2
## X-squared = 9.6087, df = 8, p-value = 0.2936
```

```r
Box.test(res2, lag = 12, type = c("Ljung-Box"), fitdf = 4)
```

```
##
##  Box-Ljung test
##
## data:  res2
## X-squared = 10.286, df = 8, p-value = 0.2455
```

```r
Box.test(res2^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res2^2
## X-squared = 6.9376, df = 12, p-value = 0.8617
```

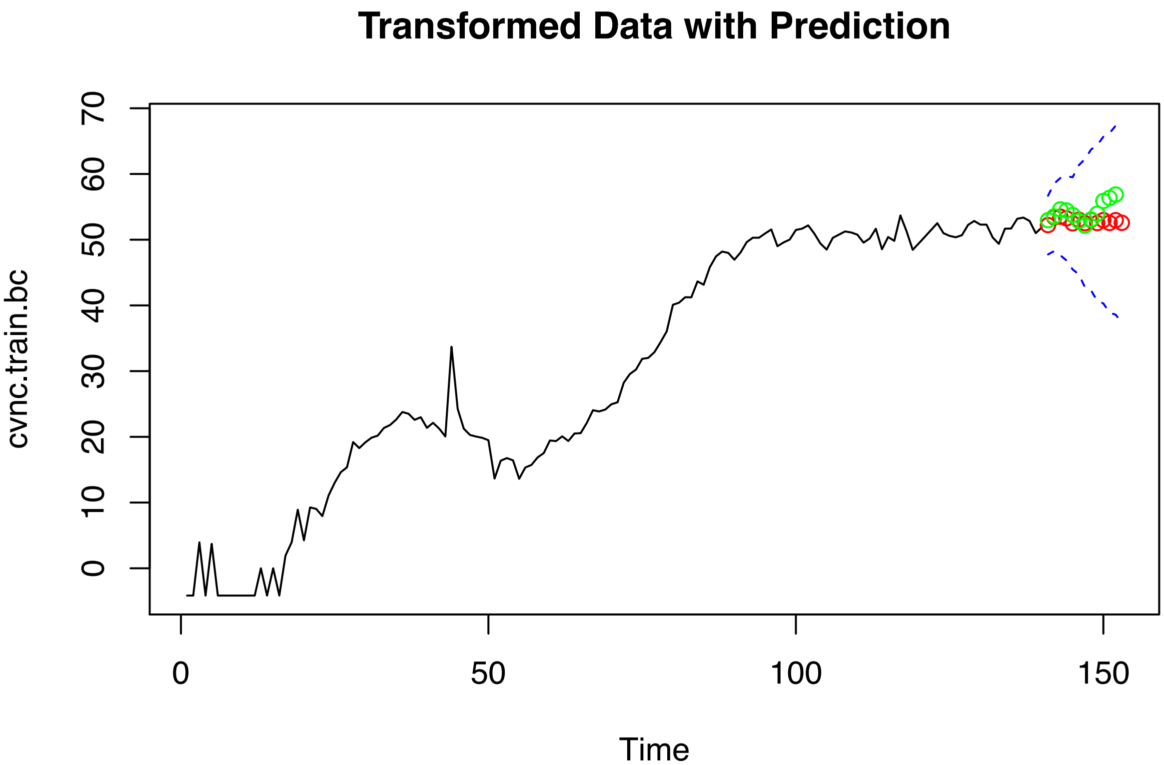
```r
ar(res2, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res2, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  5.187
```

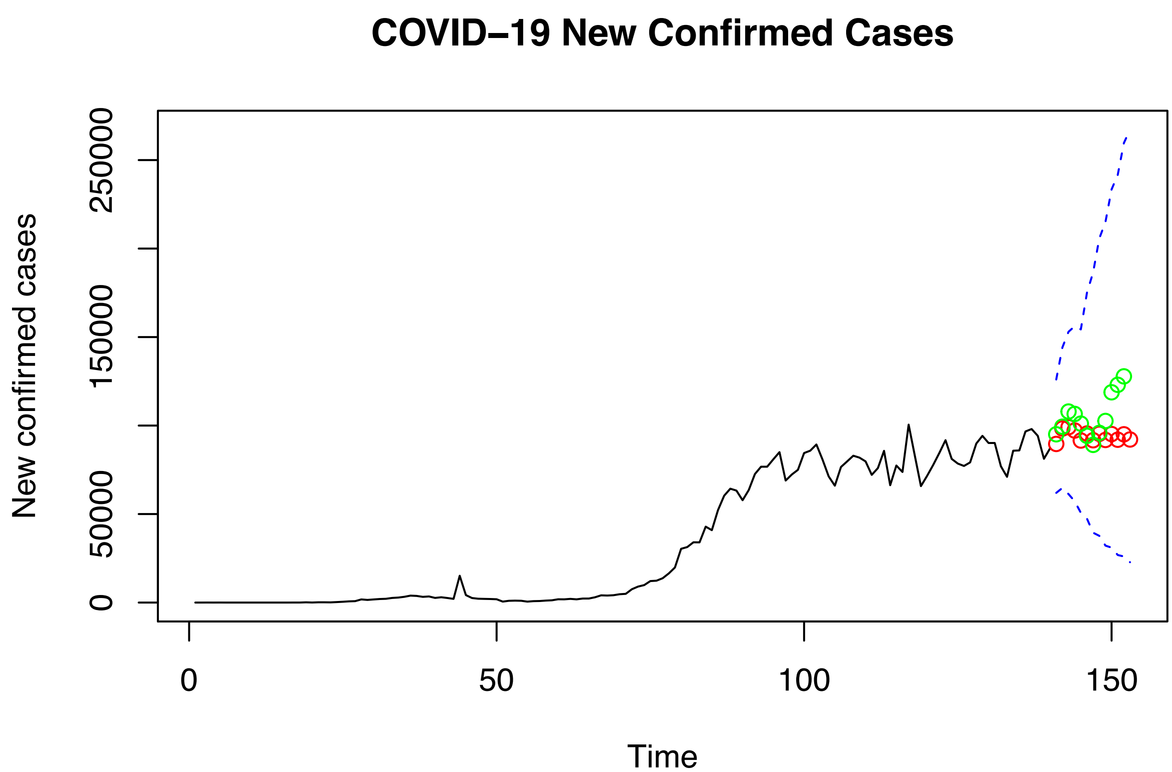
Similar with model 1, the residuals pass all tests but fail Shapiro-Wilk normality test and Q-Q plot with the reason of non-normal original time series. Besides, residuals of model 2 also have ACF at lag = 11 and 16 outside confidence interval and PACF at lag = 16 outside confidence interval. We should have applied modification to model 2 based on the model of residuals. However, since the model after modification has too many coefficients to do residual tests (degree of freedom will be computed as negative number which is unreasonable), we will abandon this model.

Thus, final model for the Box-Cox transformed time series:
ARIMA(1, 1, 6) model:
$(1+0.9575B_{(0.0411)})(1-B)X_t = (1+0.5265_{(0.0842)}B-0.3411_{(0.0692)}B^2+0.4171_{(0.1059)}B^5+0.4931_{(0.1043)}B^6)Z_t$
$\sigma^2 = 4.981$

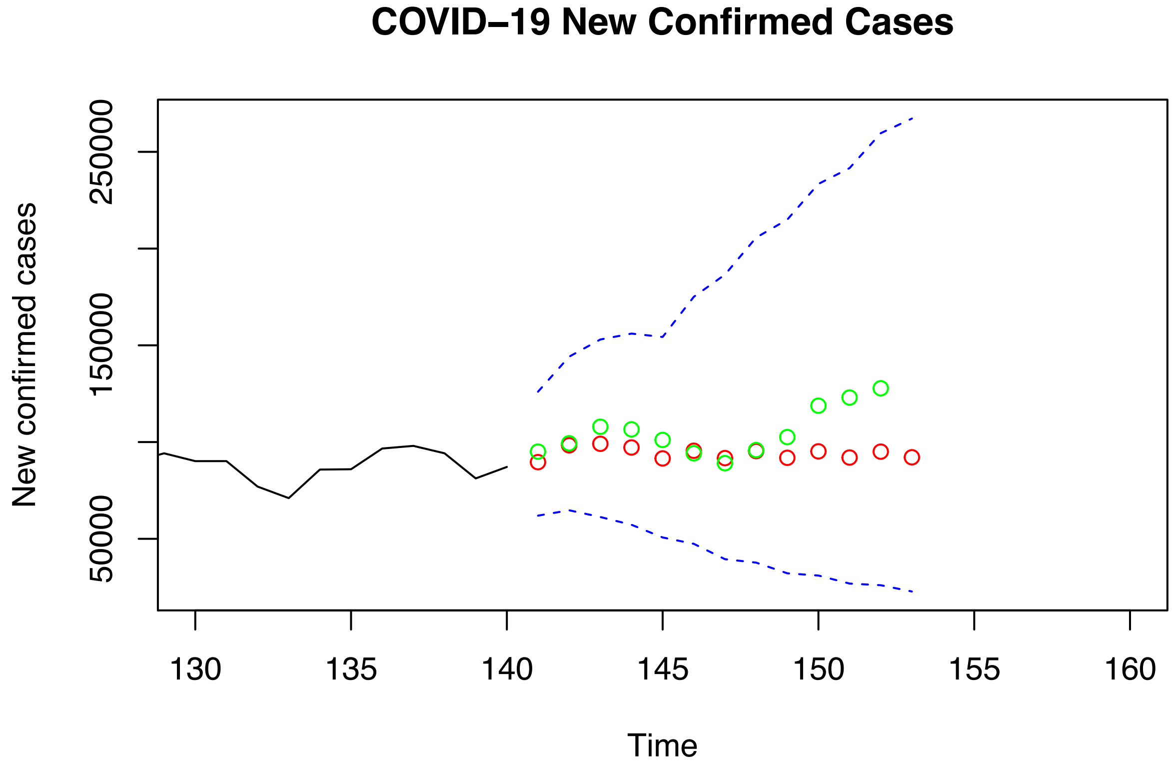
Forecast

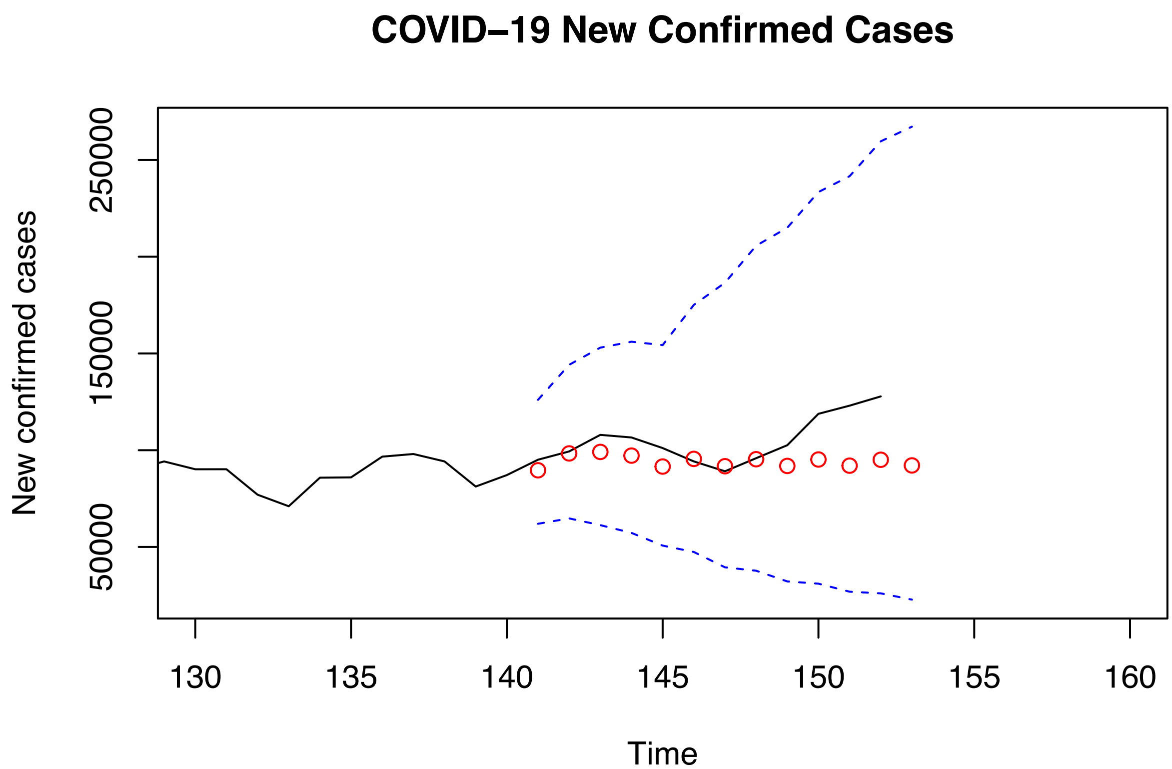


Green points are original data and red points are forecast. The small peak in the first half of the forecast is predicted but the second half is not predicted very well.



Green points are original data and red points are forecast. Similar to the prediction of the transformed data, the small peak in the first half of the forecast is predicted well but the second half is not predicted very accurately.





From the zoomed plots, we can see more clearly that the first 8 points are very similar to the real values and gives a trend similar to the real trend. But the last 5 points are slightly away from the real values.

Conclusion

The ARIMA(1, 1, 6) model $(1 + 0.9575B_{(0.0411)})(1 - B)X_t = (1 + 0.5265_{(0.0842)}B - 0.3411_{(0.0692)}B^2 + 0.4171_{(0.1059)}B^5 + 0.4931_{(0.1043)}B^6)Z_t$ is the final model for the project. Based on the forecast, the model can provide an effective prediction for new confirmed COVID-19 cases. Therefore, the goal of forecasting the future new confirmed COVID-19 cases is achieved.

The math formula used for the model is ARIMA(p, d, q) which is $\phi(B)(1 - B)^d X_t = \theta(B)Z_t$. I would like to thank professor Feldman, Angel Chen and Kerry Wang who both are classmates in PSTAT 174, for helping me with my project.

References

European Centre for Disease Prevention and Control.(2020).New Cases[Data file].Retrieved from https://github.com/owid/covid-19-data/blob/master/public/data/ecdc/new_cases.csv

Appendix

Preprocessing

```
# Import packages
library(readr)
library(MASS)
library(forecast)
```

```
dt <- read.csv("~/Desktop/pstat174/project/new_cases.csv") # Load data
cp <- dt # Make a copy of data
cp["World"][cp["World"]==0] = 0.00001 # Subtitute 0s for future analyzing
cvnc_row <- nrow(cp)
cvnc <- cp[2:cvnc_row, 3]
cvnc_ts = ts(cvnc, start = 1)
```

```
# Plot orginal time series
ts.plot(cvnc_ts, ylab = "New confirmed cases", main = "Raw Data with Mean and Trend Lines")
abline(a=mean(cvnc_ts), b = 0, col="red") # Mean line
abline(lm(cvnc_ts ~ as.numeric(1:length(cvnc_ts))), col="blue") # Trend line
```

```
# Split data
cvnc_row
```

```
## [1] 153
```

```
cvnc.train <- cvnc_ts[c(1:140)] # Training data
cvnc.test <- cvnc_ts[c(141:152)] # Test data
```

```
# Plot training data
ts.plot(cvnc.train, ylab = "New confirmed Cases", main = "Training Data with Mean and Trend Lines")
abline(a=mean(cvnc.train), b = 0, col="red") # Mean line
abline(lm(cvnc.train ~ as.numeric(1:length(cvnc.train))), col="blue") # Trend line
```

```
# Plot histogram of training data
par(mfrow=c(1, 2))
hist(cvnc.train, main = "Histogram of Training Data", xlab = "New confirmed cases")
acf(cvnc.train, main = "ACF of Training Data")
```

Transformation

```
# Find lambda for Box-Cox transformation
t <- 1:length(cvnc.train)
bcTransform <- boxcox(cvnc.train ~ t, plotit = TRUE)
```

```
# Apply Box-Cox transformation
lambda <- bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
lambda
```

```
## [1] 0.2222222
```

```
cvnc.train.bc <- (1/lambda)*(cvnc.train**lambda-1)
```

```
# Plot transformed training data
par(mfrow=c(1, 2))
```

```r
ts.plot(cvnc.train, ylab = "New confirmed cases", main = "Original Training Data")
ts.plot(cvnc.train.bc, main = "Transformed Training Data", ylab = expression(Y[t]))
```

```r
# Plot histogram of transformed data
par(mfrow=c(1, 2))
hist(cvnc.train, main = "Histogram of Original Data")
hist(cvnc.train.bc, main = "Histogram of Transformed Data")
```

Differencing

```r
# Differencing at lag = 1 once
cvnc.train1 <- diff(cvnc.train.bc, 1)
# Plot transformed data with differencing once
ts.plot(cvnc.train1, main = "Transformed Training Data with Differencing at Lag = 1 Once")
```

```r
# Compute variance
var(cvnc.train1)
```

```
## [1] 6.974256
```

```r
# Differencing at lag = 1 twice
cvnc.train2 <- diff(cvnc.train1, 1)
# Plot transformed data with differencing twice
ts.plot(cvnc.train2, main = "Training Data with Differencing at Lag = 1 Twice")
```

```r
# Compute variance
var(cvnc.train2)
```

```
## [1] 20.26478
```

```r
# Plot data after transformation and differencing
par(mfrow=c(2, 2))

ts.plot(cvnc.train, ylab = "New confirmed cases", main = "Original Data")
abline(a=mean(cvnc.train), b = 0, col="red") # Mean line
abline(lm(cvnc.train ~ as.numeric(1:length(cvnc.train))), col="blue") # Trend line

ts.plot(cvnc.train.bc, main = "Box-Cox Transformed Data", ylab = expression(Y[t]))
abline(a=mean(cvnc.train.bc), b = 0, col="red") # Mean line
abline(lm(cvnc.train.bc ~ as.numeric(1:length(cvnc.train.bc))), col="blue") # Trend line

ts.plot(cvnc.train1, ylab = "Transformed_1", main = "De-trended Data")
abline(a=mean(cvnc.train1), b = 0, col="red") # Mean line
abline(lm(cvnc.train1 ~ as.numeric(1:length(cvnc.train1))), col="blue") # Trend line
```

```r
# Plot acf for data after transformation and differencing
par(mfrow=c(2, 2))
acf(cvnc.train, main = "ACF of Original Data")
acf(cvnc.train.bc, main = "ACF of Transformed Data")
acf(cvnc.train1, main = "ACF of De-trended Data")
```

```r
# Plot histogram for data after transformation and differencing
par(mfrow=c(2, 2))
hist(cvnc.train, main = "Histogram of Original Data")
hist(cvnc.train.bc, main = "Histogram of Transformed Data")
```

```r
hist(cvnc.train1, main = "Histogram of De-trended Data")
```

Model selection

```r
par(mfrow=c(1, 2))
acf(cvnc.train1, main = "ACF of De-trended Data")
pacf(cvnc.train1, main = "PACF of De-trended Data")
```

ACF outside confidence intervals: lags 1, 2, 8, 11 and 16
PACF outside confidence intervals: lags 1 and 6

```r
# Pure moving average models
arima(cvnc.train.bc, order=c(0,1,2), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 2), method = "ML")
##
## Coefficients:
##           ma1     ma2
##       -0.3814  0.1672
## s.e.   0.0853  0.0720
##
## sigma^2 estimated as 5.919:  log likelihood = -320.9,  aic = 647.8
```

```r
arima(cvnc.train.bc, order=c(0,1,8), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5     ma6      ma7     ma8
##       -0.4727  0.1770  -0.0805  0.1679  0.2477  0.1896  -0.0711  0.0846
## s.e.   0.0884  0.1053   0.1027  0.1043  0.1180  0.0973   0.1118  0.1142
##
## sigma^2 estimated as 5.135:  log likelihood = -311.87,  aic = 641.74
```

```r
arima(cvnc.train.bc, order=c(0,1,11), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 11), method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5     ma6      ma7     ma8
##       -0.4787  0.2256  -0.0626  0.1383  0.2091  0.1478  -0.0963  0.1890
## s.e.   0.0927  0.0937   0.1020  0.1029  0.1118  0.1168   0.1209  0.1131
##           ma9     ma10     ma11
##       -0.0139  -0.1056  -0.0743
## s.e.   0.0941   0.1242   0.1293
##
## sigma^2 estimated as 4.955:  log likelihood = -309.76,  aic = 643.51
```

```r
arima(cvnc.train.bc, order=c(0,1,16), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 16), method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5     ma6      ma7     ma8
##       -0.4943  0.1031  -0.0786  0.0720  0.1972  0.0575  -0.0247  0.1989
## s.e.   0.1921  0.1062   0.1133  0.1659  0.1670  0.1232   0.1100  0.1005
##           ma9    ma10     ma11    ma12    ma13    ma14    ma15    ma16
##       -0.0298  0.0268  -0.1620  0.1846  0.0000  0.2032  0.0207  0.2677
## s.e.   0.1351  0.1516   0.1251  0.1358  0.1423  0.1124  0.1405  0.1697
##
## sigma^2 estimated as 4.348:  log likelihood = -303.28,  aic = 640.56
```

```r
# Pure autoregressive models
arima(cvnc.train.bc, order=c(1,1,0), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 0), method = "ML")
##
## Coefficients:
##          ar1
##       -0.407
## s.e.   0.077
##
## sigma^2 estimated as 5.897:  log likelihood = -320.64,  aic = 645.28
```

```r
arima(cvnc.train.bc, order=c(6,1,0), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 0), method = "ML")
##
## Coefficients:
##           ar1     ar2     ar3     ar4     ar5     ar6
##       -0.4135  0.0285  0.0165  0.0898  0.1833  0.2547
## s.e.   0.0831  0.0902  0.0892  0.0900  0.0927  0.0893
##
## sigma^2 estimated as 5.447:  log likelihood = -315.37,  aic = 644.73
```

```r
# ARIMA models
arima(cvnc.train.bc, order=c(1,1,2), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 2), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2
##       -0.5684  0.1934  -0.0042
## s.e.   0.4952  0.5199   0.2282
##
## sigma^2 estimated as 5.865:  log likelihood = -320.28,  aic = 648.55
```

```r
arima(cvnc.train.bc, order=c(1,1,8), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 8), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2     ma3     ma4     ma5     ma6     ma7
##       -0.9571  0.5208  -0.3097  0.1222  0.0890  0.3881  0.4522  0.0865
## s.e.   0.0408  0.1008   0.1002  0.1030  0.1122  0.1110  0.1179  0.1137
##          ma8
##       0.1069
## s.e.  0.1070
##
## sigma^2 estimated as 4.909:  log likelihood = -309.28,  aic = 638.56
```

```r
arima(cvnc.train.bc, order=c(1,1,11), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 11), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2     ma3     ma4     ma5     ma6     ma7
##       -0.9500  0.5764  -0.2857  0.1486  0.0928  0.3396  0.3947  0.0300
## s.e.   0.0338  0.1002   0.0970  0.0997  0.1019  0.0999  0.1063  0.1082
##          ma8     ma9     ma10    ma11
##       0.1239  0.1450  -0.0644  -0.275
## s.e.  0.0976  0.1284   0.0901   0.091
##
## sigma^2 estimated as 4.442:  log likelihood = -305.01,  aic = 636.01
```

```r
arima(cvnc.train.bc, order=c(1,1,16), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 16), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2     ma3     ma4     ma5     ma6     ma7
##       -0.9065  0.4624  -0.3446  0.0772  0.0198  0.2909  0.3742  0.0295
## s.e.   0.0717  0.1182   0.1059  0.1029  0.1096  0.1125  0.1188  0.1102
##          ma8     ma9     ma10     ma11    ma12    ma13    ma14    ma15
##       0.1887  0.2128  -0.0216  -0.1699  0.0614  0.1016  0.1606  0.1891
## s.e.  0.1140  0.1205   0.1113   0.1087  0.1221  0.1123  0.0982  0.1375
##         ma16
##       0.2151
## s.e.  0.1040
##
## sigma^2 estimated as 4.101:  log likelihood = -299.82,  aic = 635.63
```

```r
arima(cvnc.train.bc, order=c(6,1,2), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 2), method = "ML")
```

```
##
## Coefficients:
##           ar1     ar2     ar3     ar4     ar5     ar6     ma1      ma2
##       -0.5329  0.4873  0.2118  0.0598  0.2043  0.2615  0.1017  -0.5755
## s.e.   0.1500  0.1552  0.1110  0.1050  0.1040  0.0950  0.1383   0.1216
##
## sigma^2 estimated as 5.171:  log likelihood = -312,  aic = 642.01
```

```r
arima(cvnc.train.bc, order=c(6,1,8), method="ML")
```

```
## Warning in arima(cvnc.train.bc, order = c(6, 1, 8), method = "ML"):
## possible convergence problem: optim gave code = 1
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 8), method = "ML")
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ar5     ar6      ma1     ma2
##       0.5250  -0.1790  -0.1732  0.0896  -0.4371  0.6945  -1.0093  0.6946
## s.e.  0.1335   0.1109   0.1154  0.1183   0.0947  0.1075   0.1616  0.1975
##           ma3     ma4     ma5      ma6     ma7     ma8
##       -0.1735  0.0472  0.5817  -0.8527  0.2784  0.2023
## s.e.   0.2090  0.1657  0.1575   0.1705  0.1571  0.1094
##
## sigma^2 estimated as 4.278:  log likelihood = -303.16,  aic = 636.33
```

```r
arima(cvnc.train.bc, order=c(6,1,11), method="ML")
```

```
## Warning in arima(cvnc.train.bc, order = c(6, 1, 11), method = "ML"):
## possible convergence problem: optim gave code = 1
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 11), method = "ML")
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6     ma1     ma2
##       -1.2392  -0.8796  -0.9395  -0.8689  -1.2450  -0.7351  0.8398  0.5353
## s.e.   0.1480   0.1034   0.1099   0.1046   0.0889   0.1356  0.1893  0.1497
##          ma3     ma4     ma5     ma6     ma7     ma8     ma9    ma10
##       0.7358  0.6581  1.5152  0.9206  0.2480  0.6353  0.6226  0.7505
## s.e.  0.1368  0.1319  0.1576  0.2888  0.1489  0.1292  0.1541  0.1566
##         ma11
##       0.2464
## s.e.  0.1599
##
## sigma^2 estimated as 3.724:  log likelihood = -297.35,  aic = 630.69
```

```r
arima(cvnc.train.bc, order=c(6,1,16), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 16), method = "ML")
##
## Coefficients:
##           ar1      ar2     ar3     ar4     ar5     ar6     ma1     ma2
```

```
##        -0.9236  -0.4893  0.2809  0.6607  0.0876  0.1205  0.5003  0.1868
## s.e.    0.4433   0.3841  0.3370  0.3357  0.3221  0.3898  0.4355  0.3499
##            ma3      ma4     ma5     ma6     ma7     ma8      ma9    ma10
##        -0.5003  -0.5529  0.5036  0.2614  0.2470  0.1275  -0.0338  0.0826
## s.e.    0.2417   0.2953  0.2720  0.5347  0.2841  0.1773   0.1434  0.1432
##           ma11     ma12    ma13    ma14    ma15    ma16
##        -0.2072  -0.0615  0.1116  0.4159  0.2445  0.3518
## s.e.    0.1550   0.1945  0.1801  0.1571  0.1601  0.1740
##
## sigma^2 estimated as 3.578:  log likelihood = -293.86,  aic = 633.72
```

Model arima(x = cvnc.train.bc, order = c(6, 1, 11), method = "ML") has lowest AIC but the number of significant coefficients are too many to continue future analyze. Therefore, we choose model starting with second lowest AIC.

Fix models

```r
# Selection 1
arima(x = cvnc.train.bc, order = c(6, 1, 16), method = "ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 16), method = "ML")
##
## Coefficients:
##            ar1      ar2     ar3     ar4     ar5     ar6     ma1     ma2
##        -0.9236  -0.4893  0.2809  0.6607  0.0876  0.1205  0.5003  0.1868
## s.e.    0.4433   0.3841  0.3370  0.3357  0.3221  0.3898  0.4355  0.3499
##            ma3      ma4     ma5     ma6     ma7     ma8      ma9    ma10
##        -0.5003  -0.5529  0.5036  0.2614  0.2470  0.1275  -0.0338  0.0826
## s.e.    0.2417   0.2953  0.2720  0.5347  0.2841  0.1773   0.1434  0.1432
##           ma11     ma12    ma13    ma14    ma15    ma16
##        -0.2072  -0.0615  0.1116  0.4159  0.2445  0.3518
## s.e.    0.1550   0.1945  0.1801  0.1571  0.1601  0.1740
##
## sigma^2 estimated as 3.578:  log likelihood = -293.86,  aic = 633.72
```

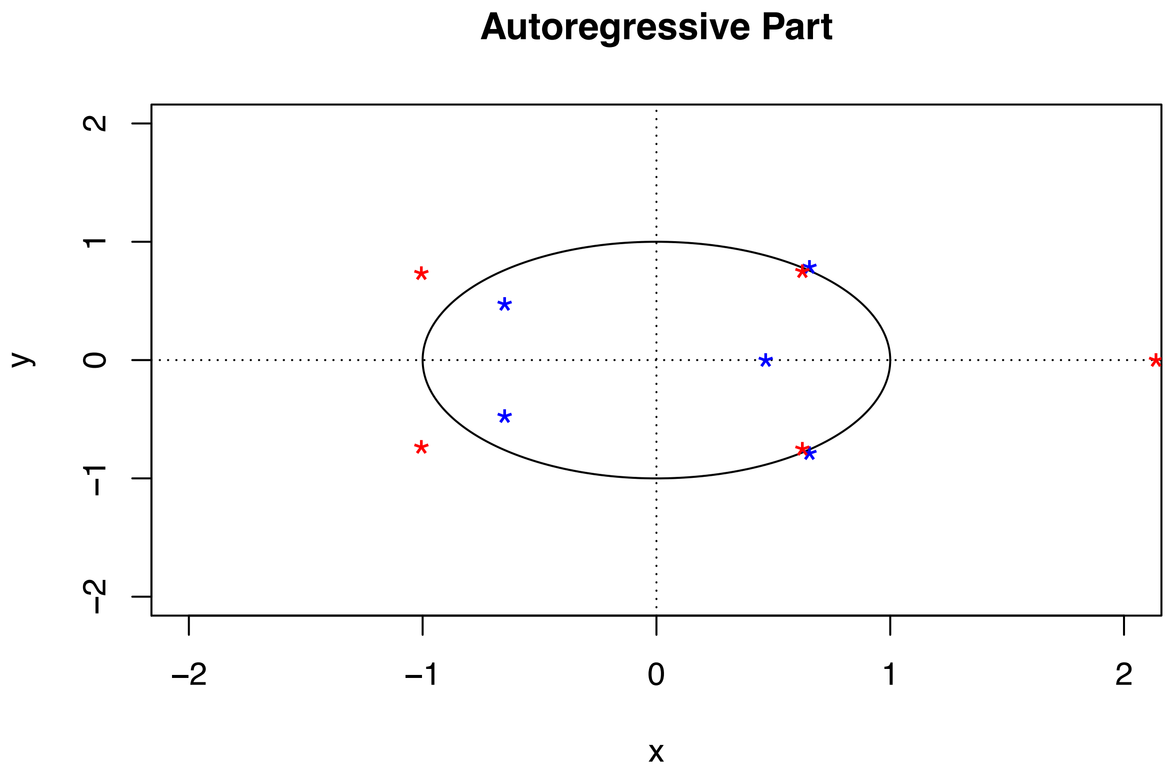
```r
# Fix coefficients
arima(x = cvnc.train.bc, order = c(5, 1, 16), method = "ML", fixed = c(NA, 0, NA, NA, NA, 0, 0, NA, 0, N
```

```
## Warning in arima(x = cvnc.train.bc, order = c(5, 1, 16), method = "ML", :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(5, 1, 16), fixed = c(NA, 0, NA, NA, NA, 0,
##     0, NA, 0, NA, 0, 0, 0, 0, 0, NA, 0, NA, NA, 0, NA), method = "ML")
##
## Coefficients:
##            ar1  ar2     ar3     ar4      ar5  ma1  ma2      ma3  ma4
##        -0.4807    0  0.5175  0.4361  -0.3167    0    0  -0.6523    0
## s.e.    0.0637    0  0.0876  0.1015   0.0926    0    0   0.0964    0
##            ma5  ma6  ma7  ma8  ma9  ma10     ma11  ma12    ma13    ma14
##         0.8097    0    0    0    0     0  -0.2781     0  0.2657  0.2967
```

```
## s.e.  0.0929    0    0    0    0     0   0.1392     0  0.1178  0.0959
##        ma15    ma16
##           0  0.2677
## s.e.      0  0.1193
##
## sigma^2 estimated as 3.273:  log likelihood = -298.01,  aic = 618.02
```

```
# Check stationarity
source("plot.roots.R")
plot.roots(NULL, polyroot(c(1, -0.4807, 0, 0.5175, 0.4361, -0.3167)), main="Autoregressive Part")
```



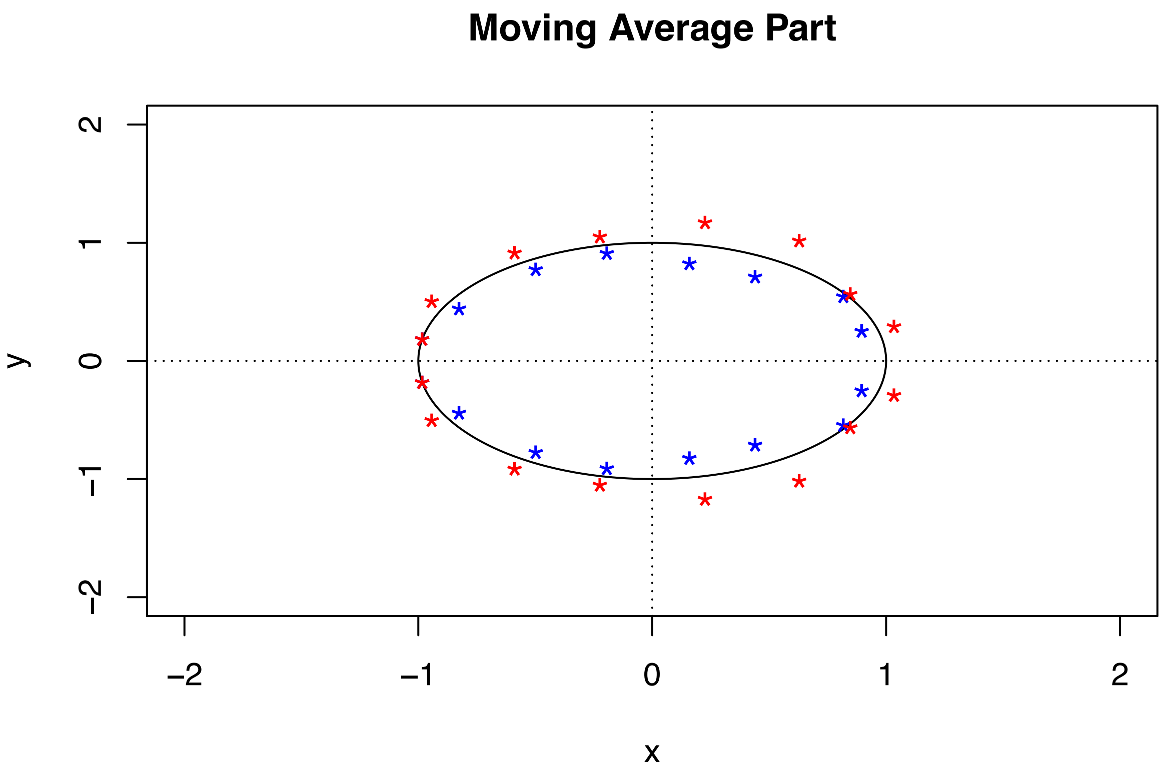

```
# Selection 2
arima(x = cvnc.train.bc, order = c(1, 1, 16), method = "ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 16), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2     ma3     ma4     ma5     ma6     ma7
##       -0.9065  0.4624  -0.3446  0.0772  0.0198  0.2909  0.3742  0.0295
## s.e.   0.0717  0.1182   0.1059  0.1029  0.1096  0.1125  0.1188  0.1102
##          ma8     ma9     ma10     ma11    ma12    ma13    ma14    ma15
##       0.1887  0.2128  -0.0216  -0.1699  0.0614  0.1016  0.1606  0.1891
## s.e.  0.1140  0.1205   0.1113   0.1087  0.1221  0.1123  0.0982  0.1375
##         ma16
##       0.2151
## s.e.  0.1040
##
## sigma^2 estimated as 4.101:  log likelihood = -299.82,  aic = 635.63
```

```r
# Fix coefficients
arima(x = cvnc.train.bc, order = c(1, 1, 16), method = "ML", fixed = c(NA, NA, NA, 0, 0, NA, NA, 0, NA,
```

```
## Warning in log(s2): NaNs produced

##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 16), fixed = c(NA, NA, NA, 0, 0, NA,
##     NA, 0, NA, 0, NA, 0, 0, NA, NA, 0, NA), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2  ma3  ma4     ma5     ma6  ma7     ma8  ma9
##       -0.8375  0.3738  -0.3525    0    0  0.3782  0.3611    0  0.1749    0
## s.e.   0.0884  0.1098   0.0838    0    0  0.0927  0.1075    0  0.0926    0
##          ma10  ma11  ma12    ma13    ma14  ma15    ma16
##       -0.0179     0     0  0.1672  0.1452     0  0.2646
## s.e.   0.0817     0     0  0.0819  0.0799     0  0.1005
##
## sigma^2 estimated as 4.241:  log likelihood = -301.78,  aic = 625.57
```

```r
# Check invertibility
plot.roots(NULL, polyroot(c(1, 0.3738, -0.3525, 0, 0, 0.3782, 0.3611, 0, 0.1749, 0, -0.0179, 0, 0, 0.16
```




```r
# Selection 3
arima(x = cvnc.train.bc, order = c(1, 1, 11), method = "ML")
```

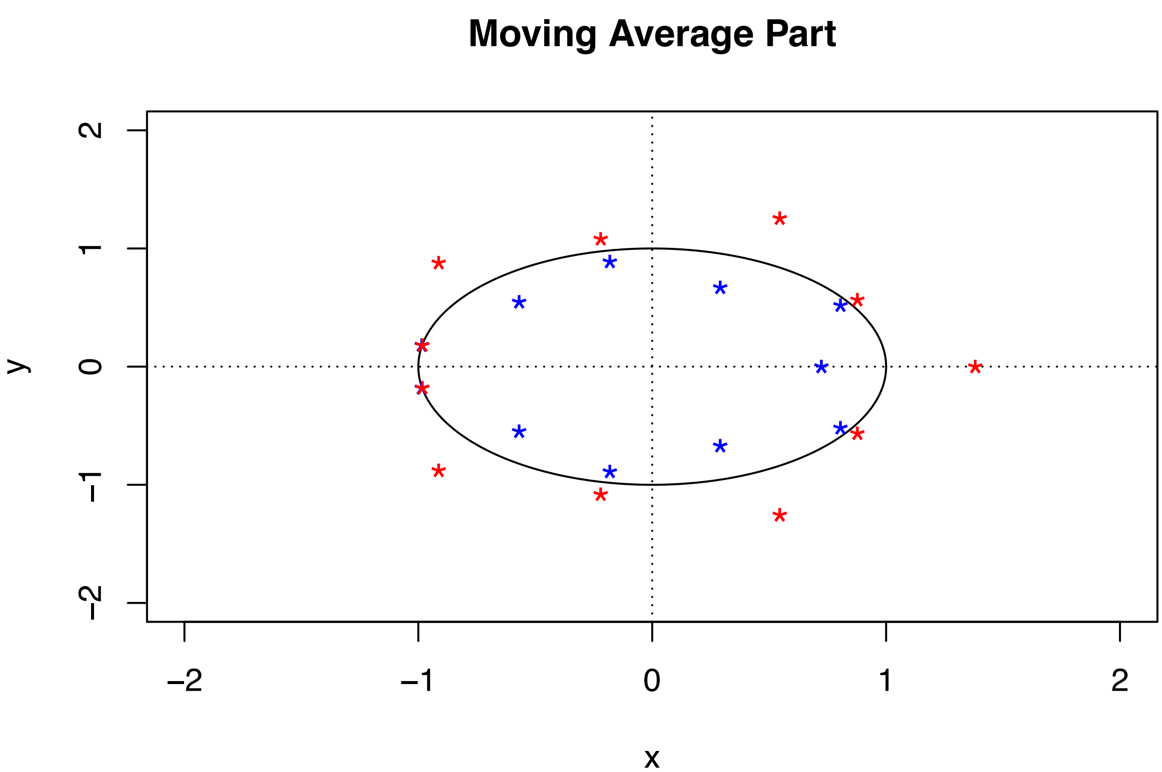
```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 11), method = "ML")
##
## Coefficients:
```

```
##           ar1     ma1      ma2     ma3     ma4     ma5     ma6     ma7
##       -0.9500  0.5764  -0.2857  0.1486  0.0928  0.3396  0.3947  0.0300
## s.e.   0.0338  0.1002   0.0970  0.0997  0.1019  0.0999  0.1063  0.1082
##          ma8     ma9     ma10    ma11
##       0.1239  0.1450  -0.0644  -0.275
## s.e.  0.0976  0.1284   0.0901   0.091
##
## sigma^2 estimated as 4.442:  log likelihood = -305.01,  aic = 636.01
```

```
# Fix coefficients
arima(x = cvnc.train.bc, order = c(1, 1, 11), method = "ML", fixed = c(NA, NA, NA, NA, 0, NA, NA, 0, 0,
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 11), fixed = c(NA, NA, NA, NA, 0, NA,
##     NA, 0, 0, 0, 0, NA), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2     ma3  ma4     ma5     ma6  ma7  ma8  ma9
##       -0.9446  0.5515  -0.3294  0.0388    0  0.4063  0.4455    0    0    0
## s.e.   0.0357  0.0938   0.0718  0.0801    0  0.0921  0.0966    0    0    0
##       ma10     ma11
##          0  -0.1821
## s.e.     0   0.0630
##
## sigma^2 estimated as 4.549:  log likelihood = -306.63,  aic = 629.27
```

```
# Check invertibility
plot.roots(NULL, polyroot(c(1, 0.5515, -0.3294, 0.0388, 0, 0.4063, 0.4455, 0, 0, 0, 0, -0.1821)), main=
```



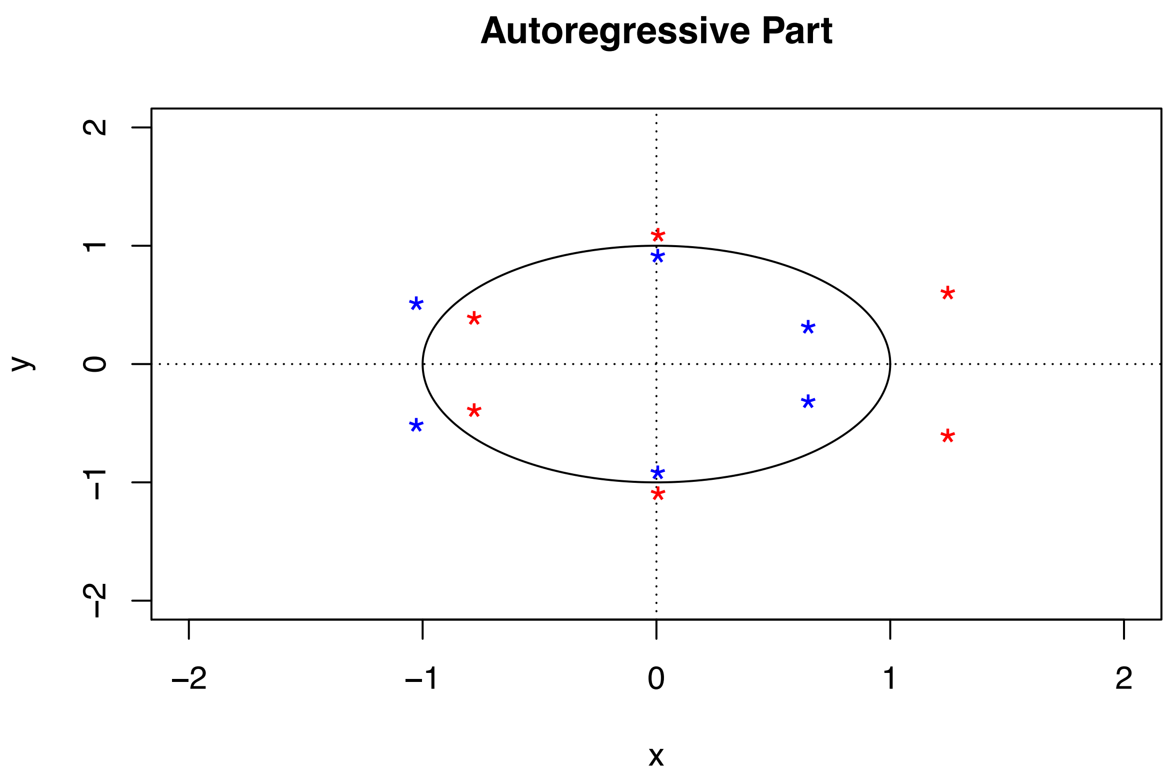
```r
# Selection 4
arima(x = cvnc.train.bc, order = c(6, 1, 8), method = "ML")
```

```
## Warning in arima(x = cvnc.train.bc, order = c(6, 1, 8), method = "ML"):
## possible convergence problem: optim gave code = 1

##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 8), method = "ML")
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ar5     ar6      ma1     ma2
##       0.5250  -0.1790  -0.1732  0.0896  -0.4371  0.6945  -1.0093  0.6946
## s.e.  0.1335   0.1109   0.1154  0.1183   0.0947  0.1075   0.1616  0.1975
##           ma3     ma4     ma5      ma6     ma7     ma8
##       -0.1735  0.0472  0.5817  -0.8527  0.2784  0.2023
## s.e.   0.2090  0.1657  0.1575   0.1705  0.1571  0.1094
##
## sigma^2 estimated as 4.278:  log likelihood = -303.16,  aic = 636.33
```

```r
# Fix coefficients
arima(x = cvnc.train.bc, order = c(6, 1, 8), method = "ML", fixed = c(NA, 0, 0, 0, NA, NA, NA, NA, NA, N
```

```
## Warning in arima(x = cvnc.train.bc, order = c(6, 1, 8), method = "ML",
## fixed = c(NA, : some AR parameters were fixed: setting transform.pars =
## FALSE

##
## Call:
## arima(x = cvnc.train.bc, order = c(6, 1, 8), fixed = c(NA, 0, 0, 0, NA, NA,
##     NA, NA, NA, NA, NA, NA, 0, NA), method = "ML")
##
## Coefficients:
##          ar1  ar2  ar3  ar4      ar5     ar6      ma1     ma2      ma3
##       0.7413    0    0    0  -0.5473  0.5757  -1.1907  0.4917  -0.3012
## s.e.  0.1667    0    0    0   0.0931  0.0952   0.1915  0.1359   0.1492
##          ma4     ma5      ma6  ma7     ma8
##       0.3040  0.6480  -0.7263    0  0.1434
## s.e.  0.1286  0.1173   0.1460    0  0.0628
##
## sigma^2 estimated as 4.426:  log likelihood = -304.1,  aic = 630.19
```

```r
# Check stationarity
plot.roots(NULL, polyroot(c(1, 0.7413, 0, 0, 0, -0.5473, 0.5757)), main="Autoregressive Part")
```



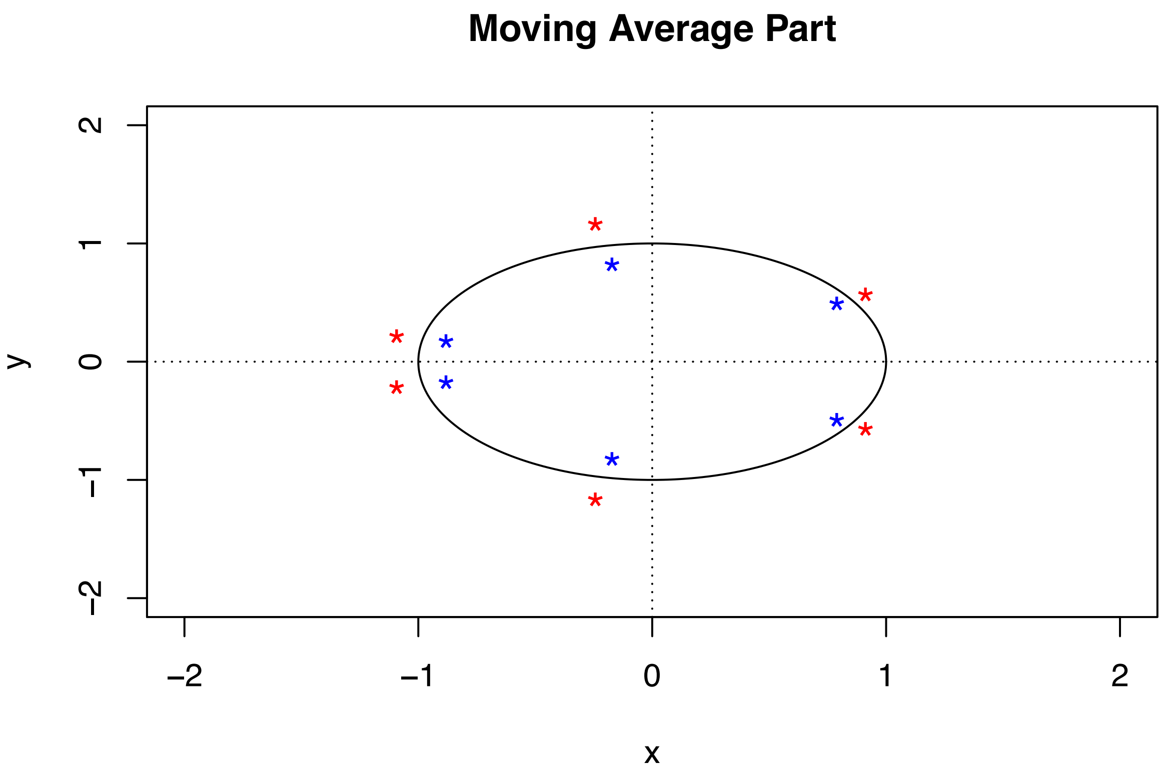

```r
# Selection 5
arima(cvnc.train.bc, order=c(1,1,8), method="ML")
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 8), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2     ma3     ma4     ma5     ma6     ma7
##       -0.9571  0.5208  -0.3097  0.1222  0.0890  0.3881  0.4522  0.0865
## s.e.   0.0408  0.1008   0.1002  0.1030  0.1122  0.1110  0.1179  0.1137
##          ma8
##       0.1069
## s.e.  0.1070
##
## sigma^2 estimated as 4.909:  log likelihood = -309.28,  aic = 638.56
```

```r
# Fix coefficients
arima(cvnc.train.bc, order=c(1,1,6), method="ML", fixed = c(NA, NA, NA, 0, 0, NA, NA))
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(1, 1, 6), fixed = c(NA, NA, NA, 0, 0, NA,
##     NA), method = "ML")
##
## Coefficients:
##           ar1     ma1      ma2  ma3  ma4     ma5     ma6
##       -0.9575  0.5265  -0.3411    0    0  0.4171  0.4931
## s.e.   0.0411  0.0842   0.0692    0    0  0.1059  0.1043
##
## sigma^2 estimated as 4.981:  log likelihood = -310.15,  aic = 632.29
```

```r
# Check invertibility
plot.roots(NULL, polyroot(c(1, 0.5265, -0.3411, 0, 0, 0.4171, 0.4931)), main="Moving Average Part")
```




```r
# Selection 6
arima(cvnc.train.bc, order=c(0,1,16), method="ML")
```

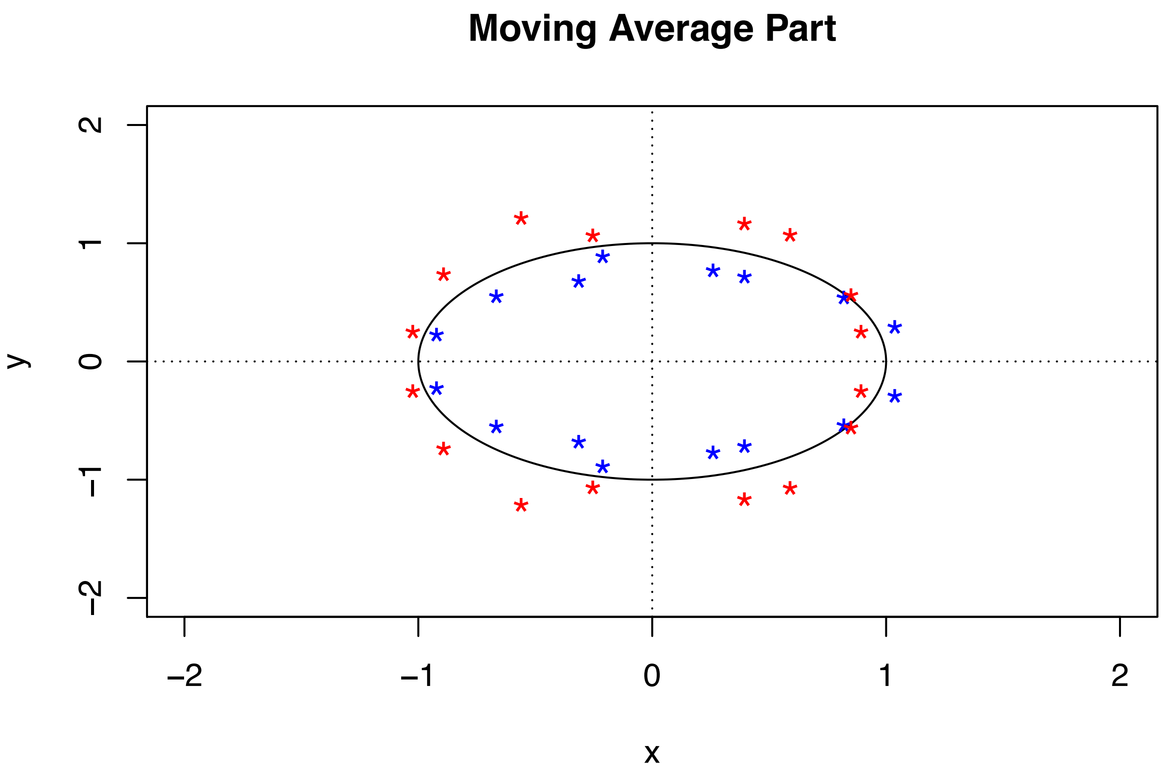
```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 16), method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5     ma6      ma7     ma8
##       -0.4943  0.1031  -0.0786  0.0720  0.1972  0.0575  -0.0247  0.1989
## s.e.   0.1921  0.1062   0.1133  0.1659  0.1670  0.1232   0.1100  0.1005
##           ma9    ma10     ma11    ma12    ma13    ma14    ma15    ma16
##       -0.0298  0.0268  -0.1620  0.1846  0.0000  0.2032  0.0207  0.2677
## s.e.   0.1351  0.1516   0.1251  0.1358  0.1423  0.1124  0.1405  0.1697
##
## sigma^2 estimated as 4.348:  log likelihood = -303.28,  aic = 640.56
```

```r
# Fix coefficients
arima(cvnc.train.bc, order=c(0,1,16), method="ML", fixed = c(NA, 0, 0, 0, NA, 0, 0, NA, 0, 0, 0, NA, 0,
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 16), fixed = c(NA, 0, 0, 0, NA, 0,
##     0, NA, 0, 0, 0, NA, 0, NA, 0, NA), method = "ML")
##
## Coefficients:
##           ma1  ma2  ma3  ma4     ma5  ma6  ma7     ma8  ma9  ma10  ma11
##       -0.8001    0    0    0  0.3163    0    0  0.2673    0     0     0
```

```
## s.e.   0.1932    0     0     0  0.1568    0     0  0.1167    0     0     0
##          ma12  ma13    ma14  ma15    ma16
##        0.3731     0  0.2879     0  0.1556
## s.e.  0.1615     0  0.0795     0  0.1893
##
## sigma^2 estimated as 3.409:  log likelihood = -305.23,  aic = 624.45
```

```r
# Check invertibility
plot.roots(NULL, polyroot(c(1, -0.8001, 0, 0, 0, 0.3163, 0, 0, 0.2673, 0, 0, 0, 0.3731, 0, 0.2879, 0, 0
```




```r
# Selection 7
arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML")
```

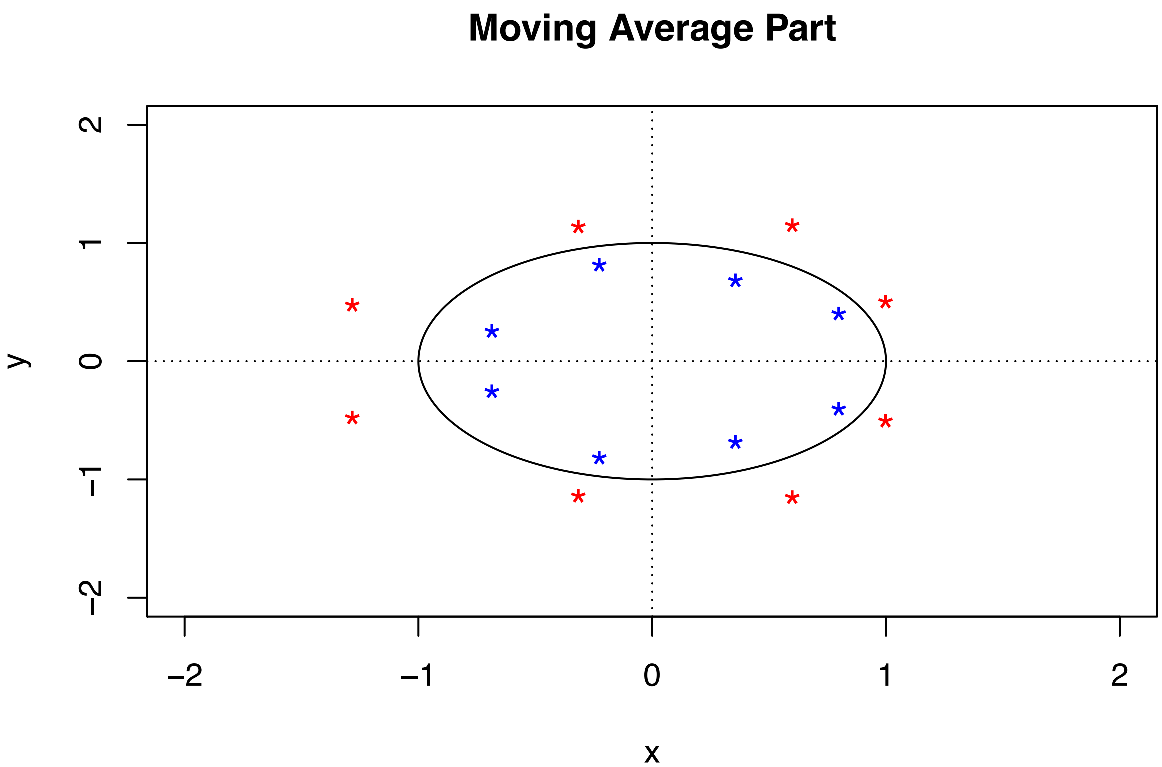
```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML")
##
## Coefficients:
##           ma1     ma2      ma3     ma4     ma5     ma6      ma7     ma8
##       -0.4727  0.1770  -0.0805  0.1679  0.2477  0.1896  -0.0711  0.0846
## s.e.   0.0884  0.1053   0.1027  0.1043  0.1180  0.0973   0.1118  0.1142
##
## sigma^2 estimated as 5.135:  log likelihood = -311.87,  aic = 641.74
```

```r
# Fix coefficients
arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML", fixed = c(NA, NA, 0, 0, NA, 0, 0, NA))
```

```
##
## Call:
## arima(x = cvnc.train.bc, order = c(0, 1, 8), fixed = c(NA, NA, 0, 0, NA, 0,
##     0, NA), method = "ML")
##
```

```
## Coefficients:
##           ma1     ma2  ma3  ma4     ma5  ma6  ma7     ma8
##       -0.4873  0.1936    0    0  0.2386    0    0  0.1819
## s.e.   0.0768  0.0903    0    0  0.1245    0    0  0.0850
##
## sigma^2 estimated as 5.312:  log likelihood = -313.78,  aic = 637.57
```

```r
# Check invertibility
plot.roots(NULL, polyroot(c(1, -0.4873, 0.1936 , 0, 0, 0.2386, 0, 0, 0.1819)), main="Moving Average Part
```



Final model chosen:
Model 1: $(1 + 0.9575B_{(0.0411)})(1 - B)X_t = (1 + 0.5265_{(0.0842)}B - 0.3411_{(0.0692)}B^2 + 0.4171_{(0.1059)}B^5 + 0.4931_{(0.1043)}B^6)Z_t$
Model 2: $(1 - B)X_t = (1 - 0.4873_{(0.0768)}B + 0.1936_{(0.0903)}B^2 + 0.2386_{(0.1245)}B^5 + 0.1819_{(0.0850)}B^8)Z_t$

##Diagnostic checking

```r
# Model 1
fit1 <- arima(cvnc.train.bc, order=c(1,1,6), method="ML", fixed = c(NA, NA, NA, 0, 0, NA, NA))
res1 <- residuals(fit1)

par(mfrow=c(2, 2))

# Plot histogram of res1
hist(res1, main = "Histogram of Residuals of Model 1")

# Plot res1
ts.plot(res1, main = "Residuals of Model 1")
abline(a=mean(res1), b = 0, col="red") # Mean line
abline(lm(res1 ~ as.numeric(1:length(res1))), col="blue") # Trend line
```

```r
# Q-Q plot of res1
qqnorm(res1)
qqline(res1, col = "Blue")

# Compute average
mean(res1)
```

```
## [1] 0.3789728
```

```r
# Plot acf and pacf for res1
par(mfrow=c(1, 2))
acf(res1, main = "ACF of Residuals of Model 1")
pacf(res1, main = "PACF of Residuals of Model 1")
```

```r
# Compute approximate value of lag
sqrt(length(cvnc.train))
```

```
## [1] 11.83216
```

```r
# Residual tests for residuals of model 1
shapiro.test(res1)
Box.test(res1, lag = 12, type = c("Box-Pierce"), fitdf = 5)
Box.test(res1, lag = 12, type = c("Ljung-Box"), fitdf = 5)
Box.test(res1^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
ar(res1, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```r
# Model 2
fit2 <- arima(x = cvnc.train.bc, order = c(0, 1, 8), method = "ML", fixed = c(NA, NA, 0, 0, NA, 0, 0, N
res2 <- residuals(fit2)

par(mfrow=c(2, 2))

# Plot histogram of res2
hist(res2, main = "Histogram of Residuals of Model 2")

# Plot res2
ts.plot(res2, main = "Residuals of Model 2")
abline(a=mean(res2), b = 0, col="red") # Mean line
abline(lm(res2 ~ as.numeric(1:length(res2))), col="blue") # Trend line

# Q-Q plot of res2
qqnorm(res2)
qqline(res2, col = "Blue")

# Compute average
mean(res2)
```

```
## [1] 0.3530809
```

```r
# Plot acf and pacf for res2
par(mfrow=c(1, 2))
acf(res2, main = "ACF of Residuals of Model 2")
pacf(res2, main = "ACF of Residuals of Model 2")
```

```r
# Residual tests for residuals of model 2
shapiro.test(res2)
Box.test(res2, lag = 12, type = c("Box-Pierce"), fitdf = 4)
```

```r
Box.test(res2, lag = 12, type = c("Ljung-Box"), fitdf = 4)
Box.test(res2^2, lag = 12, type = c("Ljung-Box"), fitdf = 0)
ar(res2, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

Forecast

```r
# Forecast with model 1
forecast(fit1, 13)
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 141       52.22152 49.36121 55.08183 47.84705 56.59599
## 142       53.40196 50.11102 56.69290 48.36890 58.43502
## 143       53.50284 49.73618 57.26950 47.74223 59.26345
## 144       53.25436 49.14766 57.36107 46.97370 59.53503
## 145       52.48988 47.99633 56.98343 45.61759 59.36217
## 146       53.02850 47.71415 58.34285 44.90090 61.15610
## 147       52.51277 46.26897 58.75658 42.96370 62.06185
## 148       53.00658 46.14058 59.87258 42.50594 63.50722
## 149       52.53376 44.93326 60.13426 40.90980 64.15773
## 150       52.98649 44.86104 61.11193 40.55970 65.41327
## 151       52.55300 43.80420 61.30181 39.17286 65.93315
## 152       52.96806 43.75463 62.18150 38.87733 67.05880
## 153       52.57064 42.80802 62.33326 37.64000 67.50128
```

```r
pred.tr <- predict(fit1, n.ahead = 13)

# Compute confidence intervals
U.tr= pred.tr$pred + 2*pred.tr$se
L.tr= pred.tr$pred - 2*pred.tr$se

#Transformation for complete data
cvnc.bc <- (1/lambda)*(cvnc**lambda-1)
```

```r
# Plot prediction on transformed data
ts.plot(cvnc.train.bc, xlim=c(1,length(cvnc.train.bc)+13), ylim = c(min(cvnc.train.bc), max(U.tr)))
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(cvnc.train.bc)+1):(length(cvnc.train.bc)+13), pred.tr$pred, col="red")
points((length(cvnc.train.bc)+1):(length(cvnc.train.bc)+13), cvnc.bc[(length(cvnc.train.bc)+1):(length(c
```

```r
# Compute prediction without transformation
pred.orig <- (pred.tr$pred*lambda+1)**(1/lambda)
U <- (U.tr*lambda+1)**(1/lambda)
L <- (L.tr*lambda+1)**(1/lambda)
```

```r
# Plot prediction on original data
ts.plot(cvnc.train, xlim=c(1,length(cvnc.train)+13), ylim = c(min(cvnc.train), max(U)), main = "COVID-19
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(cvnc.train)+1):(length(cvnc.train)+13), pred.orig, col="red")
points((length(cvnc.train)+1):(length(cvnc.train)+13), cvnc[(length(cvnc.train)+1):(length(cvnc.train)+
```

```r
# Plot prediction on original data
ts.plot(cvnc, ylim = c(min(cvnc.train), max(U)), main = "COVID-19 New Confirmed Cases", ylab = "New con
```

```
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(cvnc.train)+1):(length(cvnc.train)+13), pred.orig, col="red")
```

```
# Plot zoomed prediction on original data
ts.plot(cvnc.train, xlim = c(130, 160), ylim = c(min(L), max(U)), main = "COVID-19 New Confirmed Cases"
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(cvnc.train)+1):(length(cvnc.train)+13), pred.orig, col="red")
points((length(cvnc.train)+1):(length(cvnc.train)+13), cvnc[(length(cvnc.train)+1):(length(cvnc.train)+
```

```
# Plot zoomed prediction on original data
ts.plot(cvnc, xlim = c(130, 160), ylim = c(min(L), max(U)), main = "COVID-19 New Confirmed Cases", ylab
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(cvnc.train)+1):(length(cvnc.train)+13), pred.orig, col="red")
```